Guida per gli sviluppatori, versione 2

AWS IoT Greengrass

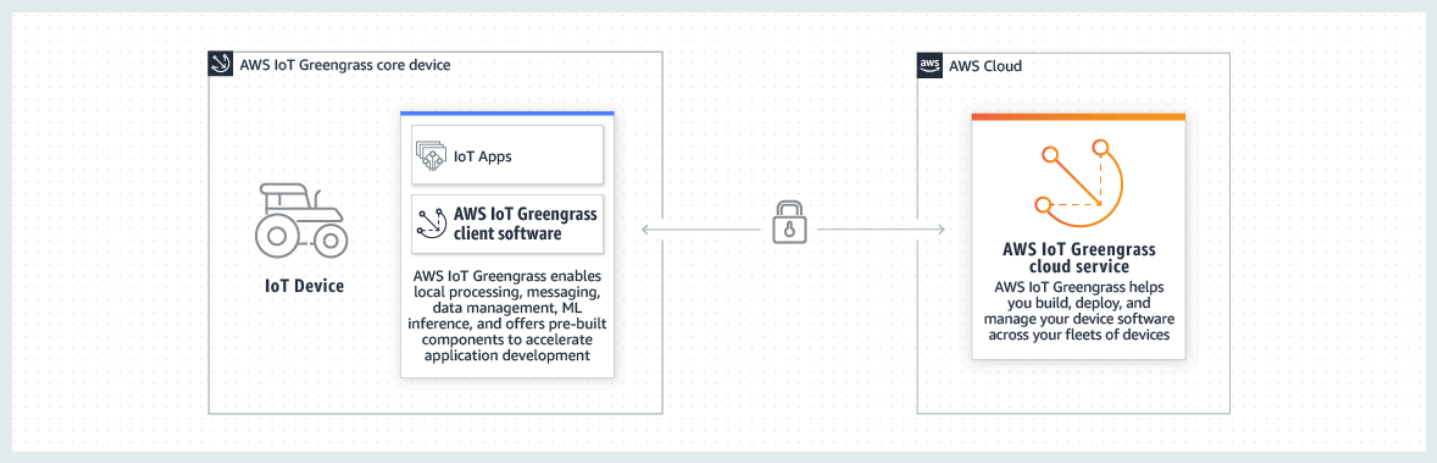
AWS IoT Greengrass: Guida per gli sviluppatori, versione 2

Table of Contents

Cos'è AWS IoT Greengrass?

AWS IoT Greengrassè un servizio cloud e runtime edge open source per l'Internet of Things (IoT) che ti aiuta a creare, implementare e gestire applicazioni IoT sui tuoi dispositivi. Puoi utilizzarlo AWS IoT Greengrass per creare software che consenta ai tuoi dispositivi di agire localmente sui dati generati, eseguire previsioni basate su modelli di apprendimento automatico e filtrare e aggregare i dati dei dispositivi. AWS IoT Greengrassconsente ai dispositivi di raccogliere e analizzare i dati più vicino al luogo in cui vengono generati, reagire in modo autonomo agli eventi locali e comunicare in modo sicuro con altri dispositivi sulla rete locale. I dispositivi Greengrass possono anche comunicare in modo sicuro AWS IoT Core ed esportare i dati IoT verso. Cloud AWS Puoi utilizzare AWS IoT Greengrass per creare applicazioni edge utilizzando moduli software predefiniti, chiamati componenti, in grado di connettere i dispositivi edge a servizi AWS o servizi di terze parti. Puoi anche utilizzarlo AWS IoT Greengrass per creare pacchetti ed eseguire il tuo software utilizzando funzioni Lambda, contenitori Docker, processi nativi del sistema operativo o runtime personalizzati a tua scelta.

L'esempio seguente mostra come un AWS IoT Greengrass dispositivo interagisce con. Cloud AWS



Nuove funzionalità

AWS IoT Greengrass V2introduce nuove funzionalità e miglioramenti. Di seguito sono incluse ulteriori informazioni sulle nuove funzionalità offerte nella versione 2.

- Cosa c'è di nuovo in AWS IoT Greengrass Version 2

Per gli utenti alle prime armi di AWS IoT Greengrass

Se non lo conosciAWS IoT Greengrass, ti consigliamo di consultare la sezione seguente:

- Funzionamento di AWS IoT Greengrass

Successivamente, segui il tutorial introduttivo per provare le funzionalità di base diAWS IoT Greengrass. In questo tutorial, installerai il software AWS IoT Greengrass Core su un dispositivo, svilupperai un componente Hello World e impacchetterai quel componente per la distribuzione.

Per gli utenti esistenti di AWS IoT Greengrass

Agli utenti attuali diAWS IoT Greengrass V1, consigliamo i seguenti argomenti per comprendere le differenze tra Greengrass versione 1 e Greengrass versione 2 e imparare a passare dalla versione 1 alla versione 2:

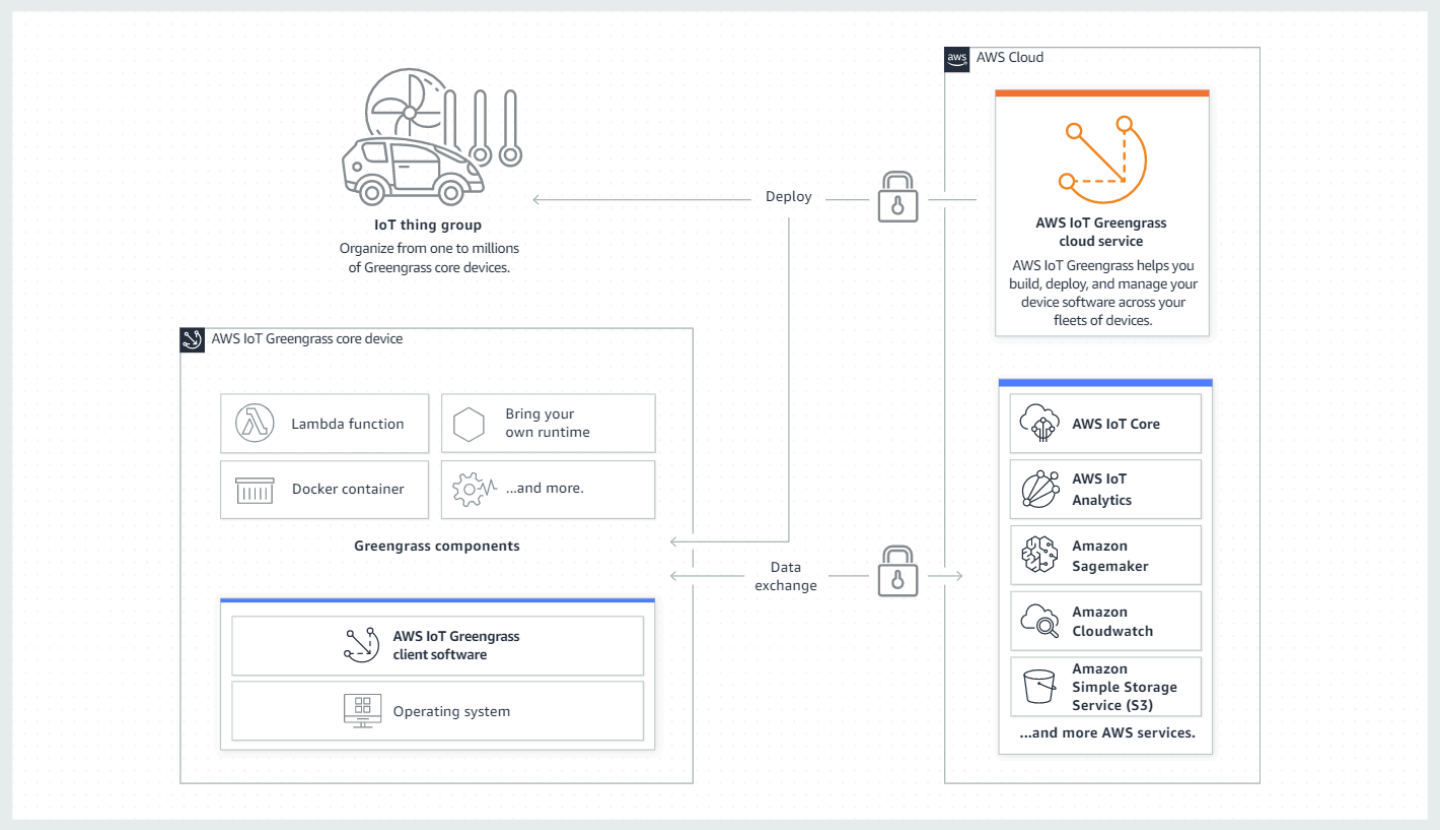
- Migrazione dalla AWS IoT Greengrass versione 1

Funzionamento di AWS IoT Greengrass

Il software AWS IoT Greengrass client, chiamato anche software AWS IoT Greengrass Core, funziona su distribuzioni basate su Windows e Linux, come Ubuntu o Raspberry Pi OS, per dispositivi con architetture ARM o x86. ConAWS IoT Greengrass, puoi programmare i dispositivi in modo che agiscano localmente sui dati generati, eseguire previsioni basate su modelli di apprendimento automatico e filtrare e aggregare i dati dei dispositivi. AWS IoT Greengrassconsente l'esecuzione locale di AWS Lambda funzioni, contenitori Docker, processi operativi nativi o runtime personalizzati a tua scelta.

AWS IoT Greengrassfornisce moduli software predefiniti denominati componenti che consentono di estendere facilmente le funzionalità dei dispositivi periferici. AWS IoT Greengrassi componenti consentono di connettersi a AWS servizi e applicazioni di terze parti sull'edge. Dopo aver sviluppato le applicazioni IoT, ti AWS IoT Greengrass consente di implementare, configurare e gestire in remoto tali applicazioni sulla tua flotta di dispositivi sul campo.

L'esempio seguente mostra come un AWS IoT Greengrass dispositivo interagisce con il servizio AWS IoT Greengrass cloud e altri AWS servizi di. Cloud AWS



Concetti chiave per AWS IoT Greengrass

Di seguito sono riportati i concetti essenziali per la comprensione e l'utilizzoAWS IoT Greengrass:

AWS IoTcosa

Qualsiasi AWS IoT cosa è una rappresentazione di un dispositivo o di un'entità logica specifica. Le informazioni su un oggetto sono memorizzate nel AWS IoT registro.

Dispositivo centrale Greengrass

Un dispositivo che esegue il software AWS IoT Greengrass Core. Un dispositivo core Greengrass è una cosa AWS IoT. Puoi aggiungere più dispositivi principali ai gruppi AWS IoT di oggetti per creare e gestire gruppi di dispositivi principali Greengrass. Per ulteriori informazioni, consulta Configurazione dei dispositivi AWS IoT Greengrass principali.

Dispositivo client Greengrass

Un dispositivo che si connette e comunica con un dispositivo core Greengrass tramite MQTT. Un dispositivo client Greengrass è una AWS IoT cosa. Il dispositivo principale può elaborare, filtrare e aggregare i dati dai dispositivi client che si connettono ad esso. È possibile configurare il dispositivo principale per inoltrare messaggi MQTT tra i dispositivi client, il servizio AWS IoT Core

cloud e i componenti Greengrass. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

I dispositivi client possono eseguire FreerTOS o utilizzare l'API di scoperta SDK per dispositivi AWS IoTo Greengrass per ottenere informazioni sui dispositivi principali a cui possono connettersi.

Componente Greengrass

Un modulo software che viene distribuito e eseguito su un dispositivo core Greengrass. Tutto il software sviluppato e distribuito con AWS IoT Greengrass è modellato come componente. AWS IoT Greengrassfornisce componenti pubblici predefiniti che forniscono caratteristiche e funzionalità utilizzabili nelle applicazioni. Puoi anche sviluppare componenti personalizzati, sul tuo dispositivo locale o nel cloud. Dopo aver sviluppato un componente personalizzato, puoi utilizzare il servizio AWS IoT Greengrass cloud per distribuirlo su dispositivi core singoli o multipli. Puoi creare un componente personalizzato e distribuirlo su un dispositivo principale. Quando lo fai, il dispositivo principale scarica le seguenti risorse per eseguire il componente:

- Ricetta: un file JSON o YAML che descrive il modulo software definendo i dettagli, la configurazione e i parametri dei componenti.
- Artefatto: il codice sorgente, i binari o gli script che definiscono il software che verrà eseguito sul dispositivo. Puoi creare artefatti partendo da zero oppure puoi creare un componente utilizzando una funzione Lambda, un contenitore Docker o un runtime personalizzato.
- Dipendenza: la relazione tra i componenti che consente di imporre aggiornamenti o riavvii automatici dei componenti dipendenti. Ad esempio, è possibile che un componente di elaborazione sicura dei messaggi dipenda da un componente di crittografia. Ciò garantisce che qualsiasi aggiornamento al componente di crittografia aggiorni e riavvii automaticamente il componente di elaborazione dei messaggi.

Per ulteriori informazioni, consultare AWS-componenti forniti e Sviluppa AWS IoT Greengrass componenti.

Distribuzione

Il processo per inviare componenti e applicare la configurazione dei componenti desiderata a un dispositivo di destinazione, che può essere un singolo dispositivo principale Greengrass o un gruppo di dispositivi principali Greengrass. Le distribuzioni applicano automaticamente tutte le configurazioni dei componenti aggiornate alla destinazione e includono tutti gli altri componenti definiti come dipendenze. È inoltre possibile clonare una distribuzione esistente per creare una nuova distribuzione che utilizza gli stessi componenti ma viene distribuita su una

destinazione diversa. Le distribuzioni sono continue, il che significa che qualsiasi aggiornamento apportato ai componenti o alla configurazione dei componenti di una distribuzione viene inviato automaticamente a tutte le destinazioni di destinazione. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.

AWS IoT GreengrassSoftware di base

L'insieme di tutti i AWS IoT Greengrass software che installi su un dispositivo principale. AWS IoT Greengrass Il software principale comprende quanto segue:

- Nucleus: questo componente richiesto fornisce le funzionalità minime del software AWS IoT Greengrass Core. Il nucleo gestisce le implementazioni, l'orchestrazione e la gestione del ciclo di vita di altri componenti. Inoltre, facilita la comunicazione tra i componenti a livello locale su un singolo dispositivo. AWS IoT Greengrass Per ulteriori informazioni, consulta Nucleo Greengrass.
- Componenti opzionali: questi componenti configurabili sono forniti AWS IoT Greengrass e abilitano funzionalità aggiuntive sui dispositivi periferici. A seconda delle esigenze, è possibile scegliere i componenti opzionali da distribuire sul dispositivo, come lo streaming di dati, l'inferenza locale dell'apprendimento automatico o un'interfaccia a riga di comando locale. Per ulteriori informazioni, consulta AWS-componenti forniti.

Puoi aggiornare il software AWS IoT Greengrass Core distribuendo nuove versioni dei componenti sul tuo dispositivo.

Caratteristiche di AWS IoT Greengrass

AWS IoT Greengrass Version 2è costituito dai seguenti elementi:

- distribuzioni software
  - Il componente Greengrass nucleus, che è l'installazione minima del AWS IoT Greengrass software Core. Questo componente gestisce le implementazioni, l'orchestrazione e la gestione del ciclo di vita dei componenti Greengrass.
  - Componenti aggiuntivi AWS forniti opzionalmente che si integrano con servizi, protocolli e software.
  - Strumenti di sviluppo Greengrass, che puoi utilizzare per creare, testare, creare, pubblicare e distribuire componenti Greengrass personalizzati.
  - IlSDK per dispositivi AWS IoT, che contiene la libreria di comunicazione interprocesso (IPC) per i componenti Greengrass personalizzati e la libreria Greengrass discovery per i dispositivi client.

  - L'SDK Stream Manager, che puoi utilizzare per gestire i flussi di dati sui dispositivi principali.
- Servizio cloud
  - API AWS IoT Greengrass V2
  - Console AWS IoT Greengrass V2

Software AWS IoT Greengrass Core

È possibile utilizzare il software AWS IoT Greengrass Core in esecuzione sui dispositivi periferici per effettuare le seguenti operazioni:

- Elabora i flussi di dati sul dispositivo locale con esportazioni automatiche nel AWS Cloud. Per ulteriori informazioni, consulta Gestisci i flussi di dati sui dispositivi core Greengrass.
- Supporta la messaggistica MQTT tra AWS IoT e componenti. Per ulteriori informazioni, consulta AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT.
- Interagisci con dispositivi locali che si connettono e comunicano tramite MQTT. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.
- Supporta la pubblicazione locale e la messaggistica di sottoscrizione tra i componenti. Per ulteriori informazioni, consulta Pubblicare/sottoscrivere messaggi locali.
- Implementa e richiama componenti e funzioni Lambda. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.
- Gestisci i cicli di vita dei componenti, ad esempio con il supporto per l'installazione e l'esecuzione degli script. Per ulteriori informazioni, consulta AWS IoT Greengrassriferimento alla ricetta del componente.
- Esegui aggiornamenti software sicuri over-the-air (OTA) del software AWS IoT Greengrass Core e dei componenti personalizzati. Per ulteriori informazioni, consultare Aggiornamento del software AWS IoT Greengrass Core (OTA) e Implementazione AWS IoT Greengrass dei componenti sui dispositivi.
- Fornisci un'archiviazione sicura e crittografata dei segreti locali e l'accesso controllato tramite componenti. Per ulteriori informazioni, consulta Gestore segreto.
- Connessioni sicure tra i dispositivi e il AWS Cloud con l'autenticazione e l'autorizzazione dei dispositivi. Per ulteriori informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass.

Puoi configurare e gestire i dispositivi principali Greengrass tramite AWS IoT Greengrass API in cui crei distribuzioni software continue. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.

Alcune funzionalità sono supportate solo su determinate piattaforme. Per ulteriori informazioni, consulta Compatibilità delle funzionalità Greengrass per sistema operativo.

Per ulteriori informazioni sulle piattaforme, i requisiti e i download supportati, consultaConfigurazione dei dispositivi AWS IoT Greengrass principali.

Scaricando questo software accetti l'Accordo di licenza del software Greengrass Core.

Compatibilità delle funzionalità Greengrass per sistema operativo

AWS IoT Greengrass supporta dispositivi che eseguono vari sistemi operativi. Alcune funzionalità sono supportate solo su determinati sistemi operativi. Utilizza le seguenti tabelle per scoprire quali funzionalità sono disponibili per ogni sistema operativo supportato. Per ulteriori informazioni sui sistemi operativi supportati, sui requisiti e su come configurare i dispositivi core Greengrass, vedere. Configurazione dei dispositivi AWS IoT Greengrass principali

Messaggistica

Funzionalità	Linux	Windows
Scambio di messaggi MQTT tra e componenti AWS IoT	Sì	Sì
Scambia messaggi locali di pubblicazione/sottoscrizione tra i componenti	Sì	Sì
Interagisci con dispositivi IoT locali tramite MQTT	Sì	Sì

Funzionalità	Linux	Windows
Interagisci con i dispositivi Modbus-RTU locali utilizzando il componente Modbus-RTU	 Sì	 No

Sicurezza

Funzionalità	Linux	Windows
Connessioni sicure con autenticazione e autorizza zione del dispositivo	 Sì	 Sì
Implementa e accedi a segreti sicuri e crittografati da AWS Secrets Manager	 Sì	 Sì
Utilizza un modulo di sicurezza hardware (HSM) per archiviare in modo sicuro la chiave privata e il certificato del dispositivo	 Sì	 No
Controlla i dispositivi principali con AWS IoT Device Defender	 Sì	 Sì
Usa AWS le credenziali per interagire con i servizi AWS	 Sì	 Sì

Installazione

Funzionalità	Linux	Windows
Installazione AWS IoT Greengrass con provisioning automatico	 Sì	 Sì
Installazione AWS IoT Greengrass con provisioning manuale	 Sì	 Sì
Installazione AWS IoT Greengrass con il provisioning del AWS IoT parco veicoli	 Sì	 Sì
Installa AWS IoT Greengras s con plugin di provisioning personalizzati	 Sì	 Sì
Esegui AWS IoT Greengras s in un contenitore Docker utilizzando un'immagine Docker predefinita	 Sì	 No

Manutenzione e aggiornamenti remoti

Funzionalità	Linux	Windows
Esegui aggiornamenti software sicuri over-the-air (OTA)	 Sì	 Sì

Funzionalità	Linux	Windows
Gestisci i dispositivi principali con AWS Systems Manager	 Sì	 No
Connect ai dispositivi principali con AWS IoT tunneling sicuro	 Sì	 No

Machine learning

Funzionalità	Linux	Windows
Esegui inferenze di apprendimento automatico utilizzando Amazon SageMaker Edge Manager	 Sì	 Sì
Esegui inferenze di apprendimento automatico utilizzando Amazon Lookout for Vision	 Sì	 No
Esegui inferenze di apprendimento automatico utilizzando DLR	 Sì	 Sì
Esegui inferenze di apprendimento automatico utilizzando TensorFlow	 Sì	 Sì

Caratteristiche dei componenti

Funzionalità	Linux	Windows
Implementa e richiama le funzioni Lambda	 Sì	 No
Esegui i contenitori Docker nei componenti	 Sì	 Sì
Elabora ed esporta flussi di dati ad alto volume utilizzando stream manager	 Sì	 Sì
Gestisci i cicli di vita dei componenti con gli script del ciclo di vita	 Sì	 Sì
Interagisci con le ombre dei dispositivi	 Sì	 Sì
Carica i log su Amazon CloudWatch Logs	 Sì	 Sì

Funzionalità	Linux	Windows
Carica dati su Amazon CloudWatch metrics utilizzando il componente CloudWatch metrics	 Sì	 Sì
Pubblica messaggi su Amazon Simple Notification Service utilizzando il componente Amazon SNS	 Sì	 No
Pubblica i dati nei flussi di distribuzione di Amazon Data Firehose utilizzando stream manager	 Sì	 Sì
Pubblica dati nei flussi di distribuzione di Amazon Data Firehose utilizzando il componente Firehose	 Sì	 No
Raccogli e agisci in base ai parametri di telemetria del sistema in tempo reale	 Sì	 Sì
Configura i limiti delle risorse di sistema per i processi dei componenti	 Sì	 No
Metti in pausa e riprendi i processi dei componenti	 Sì	 No

Funzionalità	Linux	Windows
Integrazione con l' AWS IoT SiteWise utilizzo dei componenti AWS IoT SiteWise	 Sì	 Sì
Pubblica flussi video su Amazon Kinesis Video Streams utilizzando il connettore edge per il componente Kinesis Video Streams	 Sì	 No

Sviluppo di componenti

Funzionalità	Linux	Windows
Sviluppa componenti localmente sui dispositivi principali	 Sì	 Sì
Interagisci con un dispositivo principale utilizzando la AWS IoT Greengrass CLI	 Sì	 Sì
Interagisci con un dispositivo principale utilizzando la console di debug locale	 Sì	 Sì

Funzionalità	Linux	Windows
Usa SDK per dispositivi AWS IoT for Python nei componenti personalizzati	Sì	Sì
Usa il SDK per dispositivi AWS IoT for C++ nei componenti personalizzati	Sì	Sì
Usa il SDK per dispositivi AWS IoT per Java nei componenti personalizzati	Sì	Sì

Certificazione del dispositivo

Funzionalità	Linux	Windows
Utilizzare AWS IoT Device Tester per AWS IoT Greengrass V2 convalidare i dispositivi IoT	Sì	Sì

Cosa c'è di nuovo in AWS IoT Greengrass Version 2

AWS IoT Greengrass Version 2 è una versione principale AWS IoT Greengrass che introduce le seguenti funzionalità:

- Componenti supportati da Publisher: AWS IoT Greengrass ora offre componenti supportati da Publisher. Questi componenti sono sviluppati, offerti e sottoposti a manutenzione da fornitori terzi. Per ulteriori informazioni, consulta Componenti supportati da Publisher.
- Utilizza un dispositivo Greengrass in VPC: ora è possibile utilizzare un dispositivo core Greengrass in VPC. Ciò consente di eseguire implementazioni in VPC senza accesso pubblico a Internet. Per ulteriori informazioni, consulta Gestisci un dispositivo AWS IoT Greengrass principale in VPC.
- Greengrass Testing Framework (GTF) — GTF for AWS IoT Greengrass Version 2 è ora disponibile. GTF è una raccolta di elementi costitutivi per supportare l'automazione. end-to-end Consente ai clienti AWS IoT Greengrass Version 2 interni di utilizzare lo stesso framework di test utilizzato dal team di assistenza per le modifiche qualificanti del software, l'accettazione automatica e la garanzia della qualità. Per ulteriori informazioni, consulta Greengrass Testing Framework su Github.
- Certificato PSA: le versioni di AWS IoT Greengrass nucleus 2.7.0 e successive sono ora certificate Platform Security Architecture (PSA). Per ulteriori informazioni, consulta è certificato da PSA.AWS IoT Greengrass

AWS IoT Greengrass le note di rilascio forniscono dettagli sulle AWS IoT Greengrass versioni: nuove funzionalità, aggiornamenti e miglioramenti e correzioni generali. AWS IoT Greengrass include i seguenti tipi di versioni:

- Nuove funzionalità rilasciate per AWS IoT Greengrass
- AWS IoT Greengrass Aggiornamenti software di base

Questa sezione contiene tutte le note di AWS IoT Greengrass V2 rilascio, a partire dalle più recenti, e include le principali modifiche alle funzionalità e le correzioni di bug significative. Per informazioni su ulteriori correzioni minori, consulta l'organizzazione aws-greengrass su. GitHub

Note di rilascio

- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.12.4 il 02 aprile 2024
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.12.3 il 27 marzo 2024

- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.12.2 il 15 febbraio 2024
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.12.1 l'8 dicembre 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.12.0 il 7 novembre 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.11.3 il 18 ottobre 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.11.2 il 9 agosto 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.11.1 il 21 luglio 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.11.0 il 28 giugno 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.10.3 il 21 giugno 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.10.2 il 5 giugno 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.10.1 l'11 maggio 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.10.0 il 9 maggio 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.6 il 20 aprile 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.5 il 30 marzo 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.4 il 24 febbraio 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.3 il 1° febbraio 2023
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.2 il 22 dicembre 2022
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.1 il 18 novembre 2022
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.0 il 15 novembre 2022
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.8.1 il 13 ottobre 2022
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.8.0 il 7 ottobre 2022
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.7.0 il 28 luglio 2022
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.6.0 il 27 giugno 2022
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.6 il 31 maggio 2022
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.5 il 6 aprile 2022
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.4 il 23 marzo 2022
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.3 il 6 gennaio 2022
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.2 il 3 dicembre 2021
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.1 il 23 novembre 2021
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.0 il 12 novembre 2021
- Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.4.0 il 3 agosto 2021

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.12.4 il 02 aprile 2024

Questa versione fornisce la versione 2.12.4 del componente Greengrass nucleus e gli aggiornamenti ai componenti forniti. AWS

Data di rilascio: 02 aprile 2024

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.12.4 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema per cui il nucleo entra in una condizione di stallo durante l'avvio su alcuni dispositivi Linux.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.12.3 il 27 marzo 2024

Questa versione fornisce la versione 2.12.3 del componente Greengrass nucleus e gli aggiornamenti ai componenti forniti. AWS

Data di rilascio: 27 marzo 2024

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per

il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.12.3 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema per cui il nucleo non riporta lo stato corretto del componente dopo il riavvio del nucleo e durante il ripristino del componente. • Correzioni di bug generali e miglioramenti.
Gestore delle ombre	È disponibile la versione 2.3.7 del componente shadow manager. Correzioni di bug e miglioramenti Risolve un problema per cui lo shadow manager registra periodicamente un `NullPointerException` errore durante la sincronizzazione di uno shadow manager.
Approvvigionamento della flotta	È disponibile la versione 1.2.1 del plugin per il provisioning AWS IoT della flotta. Correzioni di bug e miglioramenti Risolve un problema per cui il plug-in di provisioning della flotta è offline durante l'avvio di Greengrass nucleus. Il plug-in Fleet Provisioning ora riprova a tempo indeterminato le chiamate MQTT Connect.
Rilevatore IP	È disponibile la versione 2.1.9 del componente disk spooler. Correzioni di bug e miglioramenti Regola la fase IP acquisita in modo da inviare i log solo a livello di registro di debug.

Componente	Dettagli
Componente del broker Moquette MQTT 3.1.1	È disponibile la versione 2.3.6 del componente broker Moquette MQTT 3.1.1. Correzioni di bug e miglioramenti Correzioni di bug generali e miglioramenti.
Gestore Lambda	È disponibile la versione 2.3.3 del componente Lambda manager. Correzioni di bug e miglioramenti Correzioni di bug generali e miglioramenti.
Console di debug locale	È disponibile la versione 2.4.2 del componente della console di debug locale. Correzioni di bug e miglioramenti Correzioni di bug generali e miglioramenti.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.12.2 il 15 febbraio 2024

Questa versione fornisce la versione 2.12.2 del componente Greengrass nucleus e gli aggiornamenti ai componenti forniti. AWS

Data di rilascio: 15 febbraio 2024

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

Important

Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.12.2 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema per cui i vecchi log non venivano puliti correttam ente. • Correzioni di bug generali e miglioramenti.
Shadow Manager	È disponibile la versione 2.3.6 del componente shadow manager. Correzioni di bug e miglioramenti Risolve un problema per cui le proprietà shadow che vengono eliminate tramite Cloud AWS aggiornamenti mentre il dispositivo è offline continuano a esistere nell'ombra locale dopo il ripristino della connettività.
Lanciatore Lambda	È disponibile la versione 2.0.13 del componente lambda launcher. Correzioni di bug e miglioramenti Correzioni di bug generali e miglioramenti.

Componente	Dettagli
Disk spooler	È disponibile la versione 1.0.3 del componente disk spooler. Correzioni di bug e miglioramenti Migliora le prestazioni riutilizzando le connessioni al database.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.12.1 l'8 dicembre 2023

Questa versione fornisce la versione 2.12.1 del componente Greengrass nucleus e gli aggiornamenti dei componenti forniti. AWS

Data di rilascio: 8 dicembre 2023

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

Important

Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per

il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.12.1 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema a causa del quale il Nucleus poteva duplicare gli abbonamenti MQTT ad argomenti di distribuzione, con conseguente ulteriore registrazione e pubblicazione di file MQTT.
Autenticazione del dispositivo client	È disponibile la versione 2.4.5 del componente di autenticazione del dispositivo client. Nuove funzionalità Aggiunge il supporto per i prefissi wildcard per la selezione dei nomi degli oggetti con il parametro. `selectionRule` Correzioni di bug e miglioramenti Risolve un problema per cui i certificati non vengono aggiornati con nuove informazioni di connettività in alcuni casi.
Disk spooler	È disponibile la versione 1.0.2 del componente disk spooler. Correzioni di bug e miglioramenti Risolve un problema per cui il campo del formato del messaggio MQTT non veniva mantenuto in alcuni casi.
Bridge MQTT	È disponibile la versione 2.3.1 del componente disk spooler. Correzioni di bug e miglioramenti Risolve un problema a causa del quale il client MQTT locale entrava in un ciclo di disconnessione.

Componente	Dettagli
Gestore dello stream	È disponibile la versione 2.1.12 del componente stream manager. Correzioni di bug e miglioramenti Aggiorna l'ordine in cui vengono utilizzate le credenziali in modo che le credenziali Greengrass siano preferite AWS per le richieste di assistenza.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.12.0 il 7 novembre 2023

Questa versione fornisce la versione 2.12.0 del componente Greengrass nucleus e gli aggiornamenti ai componenti forniti. AWS

Data di rilascio: 7 novembre 2023

Aspetti salienti

- Bootstrap on rollback: AWS IoT Greengrass ora fornisce un parametro di configurazione Greengrass nucleus chiamato. `BootstrapOnRollback` Questa funzionalità consente di eseguire le fasi del ciclo di vita di bootstrap come parte di una distribuzione di rollback.

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna

la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.12.0 del Greengrass nucleus. Nuove funzionalità • Consente di eseguire le fasi del ciclo di vita di bootstrap come parte di una distribuzione di rollback.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.11.3 il 18 ottobre 2023

Questa versione fornisce la versione 2.11.3 del componente Greengrass nucleus.

Data di rilascio: 18 ottobre 2023

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

Important

Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni

patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.11.3 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema nel nucleo a causa del quale poteva avviare erroneamente un componente quando le sue dipendenze non funzionav ano. Nuove funzionalità • Aggiunge un tipo di endpoint s3 configurabile.
Gestore Lambda	È disponibile la versione 2.3.1 del componente Lambda manager. Correzioni di bug e miglioramenti • Regola i livelli di registro per determinati errori.
Console di debug locale	È disponibile la versione 2.4.0 del componente Lambda manager. Nuove funzionalità • Aggiunge la console di debug dello stream manager.
Gestore di registri	È disponibile la versione 2.3.6 del componente log manager. Correzioni di bug e miglioramenti • Regola i livelli di registro per determinati errori.

Componente	Dettagli
Gestore delle ombre	È disponibile la versione 2.3.4 del componente Shadow manager. Correzioni di bug e miglioramenti • Aggiunge il supporto per i documenti dello stato ombra nulli e vuoti.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.11.2 il 9 agosto 2023

Questa versione fornisce la versione 2.11.2 del componente Greengrass nucleus.

Data di rilascio: 9 agosto 2023

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

Important

Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.11.2 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema nel client Nucleus MQTT 5 che poteva apparire offline quando è in uso un numero elevato (> 50) di abbonamenti. • Aggiunge un nuovo tentativo per l'errore TCP del docker dial.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.11.1 il 21 luglio 2023

Questa versione fornisce la versione 2.11.1 del componente Greengrass nucleus.

Data di rilascio: 21 luglio 2023

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per

il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.11.1 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema per cui il nucleo non si avvia se un'attività di bootstrap fallisce e il file dei metadati di distribuzione è danneggiato. • Risolve un problema per cui i componenti Lambda su richiesta non vengono segnalati negli aggiornamenti sullo stato della distribuzione. • Aggiunge il supporto per gli ID duplicati delle politiche di autorizzazione.
Gestore Lambda	È disponibile la versione 2.2.11 di Lambda manager. Correzioni di bug e miglioramenti • Risolve un problema per cui la LegacySubscriptionRouter configura zione non si aggiorna quando la configurazione Lambda cambia.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.11.0 il 28 giugno 2023

Questa versione fornisce la versione 2.11.0 del componente Greengrass nucleus.

Data di rilascio: 28 giugno 2023

Aspetti salienti

- Persistent disk spooler: AWS IoT Greengrass ora fornisce un'implementazione dello spooler persistente per i messaggi spooler dai dispositivi core Greengrass verso. AWS IoT Core Questo componente memorizzerà questi messaggi in uscita su disco. Per ulteriori informazioni, consulta Spooler del disco.

- Miglioramenti alla distribuzione locale: ora puoi annullare le distribuzioni locali, impostare policy di gestione degli errori di distribuzione e ottenere informazioni dettagliate sullo stato della distribuzione.
- Miglioramenti della velocità di registrazione: le velocità di caricamento dei log per il componente log manager sono state migliorate.

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.11.0 del Greengrass nucleus. Nuove funzionalità • Consente di annullare una distribuzione locale.

Componente	Dettagli
	• Consente di configurare una politica di gestione degli errori per una distribuzione locale. • Aggiunge il supporto per un plug-in Disk Spooler.
Greengrass CLI	È disponibile la versione 2.11.0 della Greengrass CLI. Nuove funzionalità • Consente di annullare una distribuzione locale. • Consente di configurare una politica di gestione degli errori per una distribuzione locale. • Migliora la segnalazione dettagliata dello stato dell'implementazione.
Disk spooler	È disponibile la versione 1.0.0 del componente disk spooler. • Il componente disk spooler fornisce l'archiviazione persistente dei messaggi inviati dai dispositivi core Greengrass a. AWS IoT Core
Gestore dei registri	È disponibile la versione 2.3.5 del componente log manager. Miglioramenti Migliora la velocità di caricamento dei log.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.10.3 il 21 giugno 2023

Questa versione fornisce la versione 2.10.3 del componente Greengrass nucleus.

Data di rilascio: 21 giugno 2023

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.10.3 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema per cui Greengrass non sottoscrive le notifiche di distribuzione quando utilizza il provider PKCS #11.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.10.2 il 5 giugno 2023

Questa versione fornisce la versione 2.10.2 del componente Greengrass nucleus.

Data di rilascio: 5 giugno 2023

Dettagli di rilascio

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.10.2 del Greengrass nucleus. Correzioni di bug e miglioramenti • Consente l'analisi senza distinzione tra maiuscole e minuscole dei cicli di vita dei componenti. • Risolve un problema per cui la variabile di ambiente PATH non veniva ricreata correttamente. • Risolve la codifica URI del proxy per i componenti, incluso lo stream manager per i nomi utente con caratteri speciali.
Autenticazione del dispositivo client	È disponibile la versione 2.4.2 del componente di autenticazione del dispositivo client.

Componente	Dettagli
	Nuove funzionalità Aggiunge una nuova opzione di `startupTimeoutSeconds` configurazione.
Gestore Lambda	È disponibile la versione 2.2.9 del componente Lambda manager. Correzioni di bug e miglioramenti Risolve un problema per cui il numero di porta veniva danneggiato a causa di un orologio distorto.
Gestore dei registri	È disponibile la versione 2.3.4 del componente log manager. Correzioni di bug e miglioramenti • Aggiunge il supporto per l'impostazione del `periodicUploadIntervalSec` parametro su valori frazionari. Il valore minimo è 1 microsecondo. • Risolve un problema per cui il gestore dei registri non rispetta i CloudWatch `putLogEvents` limiti.
Broker MQTT 3.1 (Moquette)	È disponibile la versione 2.3.3 del componente broker MQTT 3.1 (Moquette). Nuove funzionalità Aggiunge una nuova opzione di configurazione`startupTimeoutSeconds` .
Bridge MQTT	È disponibile la versione 2.2.6 del componente bridge MQTT. Nuove funzionalità Aggiunge una nuova opzione di `startupTimeoutSeconds` configurazione.

Componente	Dettagli
Gestore dello stream	È disponibile la versione 2.1.7 del componente stream manager. Correzioni di bug e miglioramenti Risolve un problema per cui lo stream manager non riesce a leggere correttamente la configurazione del proxy.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.10.1 l'11 maggio 2023

Questa versione fornisce la versione 2.10.1 del componente Greengrass nucleus.

Data di rilascio: 11 maggio 2023

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

Important

Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per

il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.10.1 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema che poteva causare un arresto anomalo all'avvio su alcuni processori ARMv8, incluso Jetson Nano. • Greengrass non chiude più lo standard di un componente, ma ripristina il comportamento al comportamento precedente alla versione 2.10.0
Gestore dello stream	È disponibile la versione 2.1.6 del nuovo gestore di stream. Correzioni di bug e miglioramenti Risolve un problema che poteva causare un arresto anomalo all'avvio su alcuni processori ARMv8, incluso Jetson Nano.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.10.0 il 9 maggio 2023

Questa versione fornisce la versione 2.10.0 del componente Greengrass nucleus e gli aggiornamenti ai componenti forniti. AWS

Data di rilascio: 9 maggio 2023

Aspetti salienti

- Supporto MQTT5: AWS IoT Greengrass ora supporta l'invio e la ricezione di messaggi AWS IoT Core utilizzando MQTT5. Per ulteriori informazioni, consultate Pubblicare messaggi MQTT. AWS IoT Core

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.10.0 del Greengrass nucleus. Nuove funzionalità • Aggiunge `interpolateComponentConfiguration` il supporto per l'espressione regolare vuota. Greengrass ora esegue l'interpola dall'oggetto root config. • Aggiunge il supporto per MQTT5. • Aggiunge un meccanismo per caricare rapidamente i componenti del plug-in senza eseguire la scansione. • Consente a Greengrass di risparmiare spazio su disco eliminando le immagini Docker inutilizzate. Correzioni di bug e miglioramenti • Risolve un problema per cui il rollback lascia determinati valori di configurazione al loro posto da una distribuzione.

Componente	Dettagli
	• Risolve un problema a causa del quale il nucleo Greengrass esegue la convalida di AWS una sequenza di dominio in endpoint di dati e non credenziali AWS personalizzati. • Aggiorna la risoluzione delle dipendenze multigruppo per risolvere nuovamente tutte le dipendenze di gruppo tramite Cloud AWS negoziazione, anziché limitarsi alla versione attiva. Questo aggiornam ento rimuove anche il codice di errore di distribuzione. `INSTALLED _COMPONENT_NOT_FOUND` • Aggiorna il nucleo Greengrass per saltare il download delle immagini Docker quando sono già presenti localmente. • Aggiorna il nucleo Greengrass per riavviare una fase di installazione dei componenti prima della scadenza del timeout. • Correzioni e miglioramenti minori aggiuntivi.
Gestore delle ombre	È disponibile la versione 2.3.2 del nuovo gestore delle ombre. Correzioni di bug e miglioramenti Risolve un problema per cui lo shadow manager entra nello `BROKEN` stato quando il database shadow locale è danneggiato.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.6 il 20 aprile 2023

Questa versione fornisce la versione 2.9.6 del componente Greengrass nucleus.

Data di rilascio: 20 aprile 2023

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.9.6 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema per cui una distribuzione Greengrass fallisce con l'errore LAUNCH_DIRECTORY_CORRUPTED e un successivo riavvio del dispositivo non riesce ad avviare Greengrass. Questo errore può verificarsi quando si sposta il dispositivo Greengrass tra più gruppi di oggetti con distribuzioni che richiedono il riavvio di Greengrass.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.5 il 30 marzo 2023

Questa versione fornisce la versione 2.9.5 del componente Greengrass nucleus.

Data di rilascio: 30 marzo 2023

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.9.5 del Greengrass nucleus. Nuove funzionalità • Aggiunge il supporto per la verifica della firma del software Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema per cui una distribuzione fallisce quando l'area dei metadati della ricetta locale non corrisponde alla regione di lancio di

Componente	Dettagli
	Greengrass nucleus. Il nucleo di Greengrass ora rinegozia con il cloud quando ciò accade. • Risolve un problema per cui lo spooler dei messaggi MQTT si riempie e non rimuove mai i messaggi. • Correzioni e miglioramenti minori aggiuntivi.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.4 il 24 febbraio 2023

Questa versione fornisce la versione 2.9.4 del componente Greengrass nucleus.

Data di rilascio: 24 febbraio 2023

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per

il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.9.4 del Greengrass nucleus. Correzioni di bug e miglioramenti • Verifica la presenza di un messaggio nullo prima di eliminare i messaggi QOS 0. • Tronca i valori di dettaglio dello stato del lavoro se superano il limite di 1024 caratteri. • Aggiorna lo script di bootstrap per Windows per leggere correttamente il percorso principale di Greengrass se tale percorso include spazi. • Aggiorna la sottoscrizione AWS IoT Core in modo da eliminare i messaggi del client se la risposta all'abbonamento non è stata inviata. • Assicura che il nucleo carichi la configurazione dai file di backup quando il file di configurazione principale è danneggiato o mancante.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.3 il 1° febbraio 2023

Questa versione fornisce la versione 2.9.3 del componente Greengrass nucleus.

Data di rilascio: 1 febbraio 2023

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca i componenti forniti da AWS che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.9.3 del Greengrass nucleus. Correzioni di bug e miglioramenti • Assicura che gli ID client MQTT non vengano duplicati. • Aggiunge funzionalità di lettura e scrittura dei file più affidabili per evitare e ripristinare eventuali danneggiamenti. • Riprova il docker image pull per errori specifici relativi alla rete. • Aggiunge l'opzione per la connessione MQTT`noProxyAddresses` .

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.2 il 22 dicembre 2022

Questa versione fornisce la versione 2.9.2 del componente Greengrass nucleus.

Data di rilascio: 22 dicembre 2022

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.9.2 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema per cui la configurazione `interpolateComponentConfiguration` non si applica a una distribuzione in corso. • Utilizza OSHI per elencare tutti i processi secondari.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.1 il 18 novembre 2022

Questa versione fornisce la versione 2.9.1 del componente Greengrass nucleus e gli aggiornamenti dei componenti forniti. AWS

Data di rilascio: 18 novembre 2022

Aspetti salienti

- Log manager: Log Manager ora elabora e carica direttamente i file di registro attivi invece di attendere che i nuovi file vengano ruotati. Questo miglioramento riduce in modo significativo i ritardi nei log. Per ulteriori informazioni, vedere Log manager

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.9.1 del Greengrass nucleus. Correzioni di bug e miglioramenti • Aggiunge una correzione in cui Greengrass si riavvia se una distribuz ione rimuove un componente del plug-in.

Componente	Dettagli
Gestore dei registri	È disponibile la versione 2.3.0 del nuovo gestore dei registri. **Note** Si consiglia di eseguire l'aggiornamento a Greengrass nucleus 2.9.1 quando si esegue l'aggiornamento a log manager 2.3.0. Nuove funzionalità • Riduce i ritardi nei log elaborando e caricando direttamente i file di registro attivi anziché attendere la rotazione dei nuovi file. Correzioni di bug e miglioramenti • Migliora il supporto della rotazione dei log durante la rotazione di file con un nome univoco. • Correzioni e miglioramenti minori aggiuntivi.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.9.0 il 15 novembre 2022

Questa versione fornisce la versione 2.9.0 del componente Greengrass nucleus e gli aggiornamenti ai componenti forniti. AWS

Data di rilascio: 15 novembre 2022

Aspetti salienti

- Autenticazione offline: AWS IoT Greengrass ora supporta l'autenticazione offline. Puoi configurare il tuo dispositivo AWS IoT Greengrass principale in modo che i dispositivi client possano connettersi a un dispositivo principale, anche quando il dispositivo principale non è connesso al cloud. Per ulteriori informazioni, consulta Autenticazione offline.
- Distribuzioni secondarie: ora è possibile creare distribuzioni secondarie. È possibile utilizzare una sottodistribuzione per risolvere le distribuzioni non riuscite. Ogni sottodistribuzione può testare una

configurazione diversa di una distribuzione non riuscita su un sottoinsieme più piccolo di dispositivi. Per ulteriori informazioni, consulta Creare sottodistribuzioni.

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.9.0 del Greengrass nucleus. Nuove funzionalità • Aggiunge la possibilità di creare distribuzioni secondarie che riprovano le distribuzioni con un sottoinsieme più piccolo di dispositivi. Questa funzionalità crea un modo più efficiente per testare e risolvere le distribuzioni non riuscite.

Componente	Dettagli
	Correzioni di bug e miglioramenti • Migliora il supporto per i sistemi che non dispongono di `useraddgroupadd`, `eusermod`. • Correzioni e miglioramenti minori aggiuntivi.
Autenticazione del dispositivo client	È disponibile la versione 2.3.0 del componente di autenticazione del dispositivo client. Nuove funzionalità • Aggiunge il supporto per l'autenticazione offline dei dispositivi client. Con questa funzionalità, i dispositivi client possono continuare a connettersi al dispositivo principale quando il dispositivo principale non è connesso a Internet. • Aggiunge il supporto per le autorità di certificazione (CA) fornite dal cliente. Il dispositivo principale utilizza una CA fornita dal cliente come certificato principale per generare certificati broker MQTT.
Broker MQTT 5 (EMQX)	È disponibile la versione 1.2.0 del componente. Broker MQTT 5 (EMQX) Nuove funzionalità Aggiunge il supporto per le catene di certificati.
Broker Moquette MQTT	È disponibile la versione 2.3.0 del nuovo componente del broker Moquette MQTT. Nuove funzionalità Aggiunge il supporto per le catene di certificati.
Gestore segreto	È disponibile la versione 2.1.4 del nuovo gestore segreto. Correzioni di bug e miglioramenti Risolve un problema per cui i segreti memorizzati nella cache venivano rimossi quando il gestore segreto veniva distribuito e Greengrass nucleus si riavviava.

Componente	Dettagli
Gestore dello stream	È disponibile la versione 2.1.2 del nuovo gestore di stream. Correzioni di bug e miglioramenti Risolve un problema nel sistema operativo Windows che utilizza una lingua diversa dall'inglese.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.8.1 il 13 ottobre 2022

Questa versione fornisce la versione 2.8.1 del componente Greengrass nucleus.

Data di rilascio: 13 ottobre 2022

> **Note**
>
> Se si utilizza la versione 2.8.0 di Greengrass nucleus, si consiglia vivamente di eseguire l'aggiornamento alla versione 2.8.1 di Greengrass nucleus.

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi.

Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.8.1 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema a causa del quale i codici di errore di distribuzione non venivano generati correttamente dagli errori dell'API Greengrass. • Risolve un problema per cui gli aggiornamenti dello stato del parco macchine inviano informazioni imprecise quando un componente raggiunge `ERRORED` uno stato durante un'implementazione. • Risolve un problema per cui le distribuzioni non potevano essere completate quando Greengrass aveva più di 50 abbonamenti esistenti.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.8.0 il 7 ottobre 2022

Questa versione fornisce la versione 2.8.0 del componente Greengrass nucleus e la versione 1.1.0 del componente broker MQTT 5 (EMQX).

Data di rilascio: 7 ottobre 2022

Aspetti salienti

- Codici di errore di distribuzione: il Greengrass nucleus ora riporta una risposta sullo stato di integrità della distribuzione che include codici di errore dettagliati quando l'implementazione di un componente non può essere completata. Per ulteriori informazioni, consulta Codici di errore di distribuzione dettagliati.

- Stati di errore dei componenti: il nucleo di Greengrass ora riporta una risposta sullo stato di salute dei componenti che include stati di errore dettagliati quando un componente entra nello stato or. `BROKEN` `ERRORED` Per ulteriori informazioni, consulta Codici di stato dettagliati dei componenti.

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.8.0 del Greengrass nucleus. Nuove funzionalità • Aggiorna il nucleo Greengrass per segnalare una risposta sullo stato di integrità dell'implementazione che include codici di errore dettagliati in caso di problemi nell'implementazione dei componenti su un dispositivo principale. Per ulteriori informazioni, consulta Codici di errore di distribuzione dettagliati.

Componente	Dettagli
	• Aggiorna il nucleo Greengrass per segnalare una risposta sullo stato di salute dei componenti che include codici di errore dettagliati quando un componente entra nello `BROKEN` stato or. `ERRORED` Per ulteriori informazioni, consulta Codici di stato dettagliati dei componenti. • Espande i campi dei messaggi di stato per migliorare le informazioni sulla disponibilità del cloud per i dispositivi. • Migliora lo stato della flotta e la robustezza del servizio. Correzioni di bug e miglioramenti • Consente la reinstallazione di un componente danneggiato quando la sua configurazione cambia. • Risolve un problema per cui un riavvio del nucleo durante l'impleme ntazione di bootstrap causa il fallimento di una distribuzione. • Risolve un problema in Windows in cui l'installazione non riesce quando un percorso root contiene spazi. • Risolve un problema per cui l'arresto di un componente durante una distribuzione utilizza lo script di spegnimento della nuova versione. • Vari miglioramenti allo spegnimento. • Correzioni e miglioramenti minori aggiuntivi.
Broker MQTT 5 (EMQX)	È disponibile la versione 1.1.0 del componente. Broker MQTT 5 (EMQX) Nuove funzionalità • Aggiunge il supporto per le configurazioni EMQX, comprese le opzioni del broker e i plug-in. Correzioni di bug e miglioramenti • Aggiorna EMQX alla versione 4.4.9.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.7.0 il 28 luglio 2022

Questa versione fornisce la versione 2.7.0 del componente Greengrass nucleus, la versione 2.1.0 del componente stream manager e la versione 2.2.5 del componente Lambda manager.

Data di rilascio: 28 luglio 2022

Aspetti salienti

- Metriche di telemetria di Stream Manager: Stream Manager ora invia automaticamente i parametri di telemetria ad EventBridge Amazon, così puoi creare applicazioni cloud che monitorano e analizzano il volume di dati caricati dai tuoi dispositivi principali. Per ulteriori informazioni, consulta Raccogli dati di telemetria sanitaria del sistema dai dispositivi principali AWS IoT Greengrass.
- Autorità di certificazione (CA) personalizzata: ora sono supportati i certificati client firmati da una CA di certificazione personalizzata, presso la quale la CA non è registrata. AWS IoT Per ulteriori informazioni, consulta Utilizza un certificato del dispositivo firmato da una CA privata.

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.7.0 del Greengrass nucleus. Nuove funzionalità • Aggiorna il nucleo Greengrass per inviare aggiornamenti di stato al AWS IoT Greengrass cloud quando il dispositivo principale applica una distribuzione locale. • Aggiunge il supporto per i certificati client firmati da un'autorità di certificazione (CA) personalizzata, presso la quale la CA non è registrat a. AWS IoT Per utilizzare questa funzionalità, puoi impostare la nuova opzione `greengrassDataPlaneEndpoint` di configurazione `suiotdata`. Per ulteriori informazioni, consulta Utilizza un certificato del dispositivo firmato da una CA privata. Correzioni di bug e miglioramenti • Risolve un problema per cui il nucleo Greengrass ripristina un dispiegam ento in determinati scenari quando il nucleo viene fermato o riavviato. Il nucleo ora riprende il dispiegamento dopo il riavvio del nucleo. • Aggiorna il programma di installazione di Greengrass per rispettare l'`--start`argomento quando si specifica di configurare il software come servizio di sistema. • Aggiorna il comportamento di SubscribeToComponentUpdates impostare l'ID di distribuzione negli eventi in cui il nucleo ha aggiornato un componente. • Correzioni e miglioramenti minori aggiuntivi.
Gestore di stream	È disponibile la versione 2.1.0 del componente stream manager. Nuove funzionalità • Aggiorna questo componente per inviare automaticamente i parametri di telemetria ad Amazon. EventBridge Per ulteriori informazioni, consulta Raccogli dati di telemetria sanitaria del sistema dai dispositivi principali AWS IoT Greengrass. Questa funzionalità richiede la versione 2.7.0 o successiva del componente Greengrass nucleus.

Componente	Dettagli
	• Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
Gestore Lambda	È disponibile la versione 2.2.5 del componente Lambda manager. Nuove funzionalità • Aggiunge il supporto per i caratteri jolly degli argomenti MQTT nelle fonti di eventi in cui ci si iscrive ai messaggi di pubblicazione/sottoscrizione locali. Questa funzionalità richiede la versione 2.6.0 o successiva del componente Greengrass nucleus. • Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.6.0 il 27 giugno 2022

Questa versione fornisce la versione 2.6.0 del componente Greengrass nucleus, nuovi componenti forniti e AWS aggiornamenti ai componenti forniti. AWS

Data di rilascio: 27 giugno 2022

Aspetti salienti

- Wildcard negli argomenti di pubblicazione/sottoscrizione locali: ora è possibile utilizzare i caratteri jolly MQTT quando ci si abbona agli argomenti di pubblicazione/sottoscrizione locali. Per ulteriori informazioni, consultare Pubblicare/sottoscrivere messaggi locali e SubscribeToTopic.
- Supporto per le ombre dei dispositivi client: ora è possibile interagire con le ombre dei dispositivi client in componenti personalizzati e sincronizzare le ombre dei dispositivi client con. AWS IoT Core Per ulteriori informazioni, consulta Interazione e sincronizzazione delle ombre dei dispositivi client.
- Supporto MQTT 5 locale per i dispositivi client: ora è possibile implementare il broker EMQX MQTT 5 per utilizzare le funzionalità MQTT 5 nella comunicazione tra i dispositivi client e un dispositivo principale. Per ulteriori informazioni, consultare Broker MQTT 5 (EMQX) e Connect i dispositivi client ai dispositivi principali.

- Variabili di ricetta nelle configurazioni dei componenti: ora è possibile utilizzare variabili di ricetta specifiche nelle configurazioni dei componenti. È possibile utilizzare queste variabili di ricetta quando si definisce la configurazione predefinita di un componente in una ricetta o quando si configura un componente in una distribuzione. Per ulteriori informazioni, consultare Variabili di ricetta e Usa le variabili di ricetta negli aggiornamenti di fusione.
- Carte jolly nelle politiche di autorizzazione IPC: ora è possibile utilizzare i caratteri * jolly per abbinare qualsiasi combinazione di caratteri nelle politiche di autorizzazione IPC (Interprocess Communication). Questo jolly consente di consentire l'accesso a più risorse in un'unica politica di autorizzazione. Per ulteriori informazioni, consulta Wildcard nelle politiche di autorizzazione.
- Operazioni IPC che gestiscono le distribuzioni e i componenti locali: ora puoi sviluppare componenti personalizzati per gestire le distribuzioni locali e visualizzare i dettagli dei componenti. Per ulteriori informazioni, consulta IPC: gestione delle distribuzioni e dei componenti locali.
- Operazioni IPC che autenticano e autorizzano i dispositivi client: ora puoi utilizzare queste operazioni per creare un componente broker locale personalizzato. Per ulteriori informazioni, consulta IPC: autenticazione e autorizzazione dei dispositivi client.

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per

il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.6.0 del Greengrass nucleus. Nuove funzionalità • Aggiunge il supporto per le wildcard MQTT quando ci si abbona ad argomenti di pubblicazione/sottoscrizione locali. Per ulteriori informazi oni, consultare Pubblicare/sottoscrivere messaggi locali e Subscribe ToTopic. • Aggiunge il supporto per le variabili di ricetta nelle configurazioni dei componenti, diverse dalla variabile di ricetta. *component_dependen cy_name* :configuration: *json_pointer* È possibile utilizzar e queste variabili di ricette quando si definisce un componente DefaultConfiguration in una ricetta o quando si configura un componente in una distribuzione. Per abilitare questa funzionalità, imposta l'opzione interpolateComponentConfigurationdi configurazione sutrue. Per ulteriori informazioni, consultare Variabili di ricetta e Usa le variabili di ricetta negli aggiornamenti di fusione. • Aggiunge il supporto completo per la * wildcard nelle politiche di autorizzazione della comunicazione tra processi (IPC). È ora possibile specificare il * carattere in una stringa di risorse in modo che corrispon da a qualsiasi combinazione di caratteri. Per ulteriori informazioni, consulta Wildcard nelle politiche di autorizzazione. • Aggiunge il supporto per i componenti personalizzati per chiamare le operazioni IPC utilizzate dalla CLI di Greengrass. È possibile utilizzar e queste operazioni IPC per gestire le distribuzioni locali, visualizzare i dettagli dei componenti e generare una password che è possibile utilizzare per accedere alla console di debug locale. Per ulteriori informazioni, consulta IPC: gestione delle distribuzioni e dei componenti locali.

Componente	Dettagli
	Correzioni di bug e miglioramenti • Risolve un problema per cui i componenti dipendenti non reagivano quando le loro dipendenze rigide si riavviavano o cambiavano stato in determinati scenari. • Migliora i messaggi di errore che il dispositivo principale segnala al servizio AWS IoT Greengrass cloud quando un'implementazione fallisce. • Risolve un problema per cui il nucleo di Greengrass applicava due volte il dispiegamento di un oggetto in determinati scenari al riavvio del nucleo. • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.
Broker MQTT 5 (EMQX)	È disponibile la versione 1.0.0 del nuovo componente broker EMQX MQTT 5. Nuove funzionalità • Aggiunge il supporto per il broker EMQX MQTT 5 locale. I dispositivi client possono connettersi a questo broker MQTT per comunicare con un dispositivo principale utilizzando le funzionalità MQTT 5.

Componente	Dettagli
Gestore delle ombre	È disponibile la versione 2.2.0 del componente shadow manager. Nuove funzionalità • Aggiunge il supporto per il servizio shadow locale tramite l'interfaccia locale di pubblicazione/sottoscrizione. È ora possibile comunicare con il broker di messaggi di pubblicazione/sottoscrizione locale su argomenti Shadow MQTT per ottenere, aggiornare ed eliminare le ombre sul dispositivo principale. Questa funzionalità consente di connettere i dispositivi client al servizio shadow locale utilizzando il bridge MQTT per inoltrare messaggi su argomenti shadow tra i dispositivi client e l'interfa ccia locale di pubblicazione/sottoscrizione. Questa funzionalità richiede la versione 2.6.0 o successiva del componente Greengrass nucleus. Per connettere i dispositivi client al servizio shadow locale, è necessario utilizzare anche la versione 2.2.0 o successiva del componente bridge MQTT. • Aggiunge l'`direction` opzione che è possibile configurare per personalizzare la direzione di sincronizzazione delle ombre tra il servizio shadow locale e il. Cloud AWS È possibile configurare questa opzione per ridurre la larghezza di banda e le connessioni a. Cloud AWS
Autenticazione del dispositivo client	È disponibile la versione 2.2.0 del componente di autenticazione del dispositivo client. Nuove funzionalità • Aggiunge il supporto per componenti personalizzati per chiamare le operazioni di comunicazione tra processi (IPC) per autenticare e autorizzare i dispositivi client. È possibile utilizzare queste operazioni in un componente broker MQTT personalizzato, ad esempio. Per ulteriori informazioni, consulta IPC: autenticazione e autorizzazione dei dispositivi client. • Aggiunge le `threadPoolSize` opzioni `maxActiveAuthTokens` `cloudQueueSize` , e che è possibile configurare per ottimizzare le prestazioni di questo componente.

Componente	Dettagli
Bridge MQTT	È disponibile la versione 2.2.0 del componente bridge MQTT. Nuove funzionalità • Aggiunge il supporto per i caratteri jolly dell'argomento MQTT (#and+) quando si specifica local publish/subscribe come broker di messaggi di origine. Questa funzionalità richiede la versione 2.6.0 o successiva del componente Greengrass nucleus. • Aggiunge l'`targetTopicPrefix` opzione, che è possibile specificare per configurare il bridge MQTT per aggiungere un prefisso all'argomento di destinazione quando inoltra un messaggio.
Greengrass CLI	È disponibile la versione 2.6.0 della Greengrass CLI. Nuove funzionalità • Aggiunge il supporto per i componenti personalizzati per chiamare le operazioni di comunicazione tra processi (IPC) utilizzate dalla CLI Greengrass. È possibile utilizzare queste operazioni IPC per gestire le distribuzioni locali, visualizzare i dettagli dei componenti e generare una password da utilizzare per accedere alla console di debug locale. Per ulteriori informazioni, consulta IPC: gestione delle distribuzioni e dei componenti locali. Correzioni di bug e miglioramenti • Correzioni e miglioramenti minori aggiuntivi.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.6 il 31 maggio 2022

Questa versione fornisce la versione 2.5.6 del componente Greengrass nucleus e la versione 2.2.4 del componente log manager.

Data di rilascio: 31 maggio 2022

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.5.6 del Greengrass nucleus. Nuove funzionalità • Aggiunge il supporto per i moduli di sicurezza hardware che utilizzan o chiavi ECC. È possibile utilizzare un modulo di sicurezza hardware (HSM) per archiviare in modo sicuro la chiave privata e il certificato del dispositivo. Per ulteriori informazioni, consulta Integrazione della sicurezza hardware. Correzioni di bug e miglioramenti • Risolve un problema per cui la distribuzione non viene mai completata quando si distribuisce un componente con uno script di installazione non funzionante in determinati scenari.

Componente	Dettagli
	• Migliora le prestazioni durante l'avvio. • Correzioni e miglioramenti minori aggiuntivi.
Gestore dei registri	È disponibile la versione 2.2.4 del componente log manager. Correzioni di bug e miglioramenti • Migliora la stabilità nella gestione di configurazioni non valide. • Correzioni e miglioramenti minori aggiuntivi.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.5 il 6 aprile 2022

Questa versione fornisce la versione 2.5.5 del componente Greengrass nucleus.

Data di rilascio: 6 aprile 2022

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per

il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.5.5 del Greengrass nucleus. Nuove funzionalità • Aggiunge la variabile di `GG_ROOT_CA_PATH` ambiente per i componenti, in modo da poter accedere al certificato dell'autorità di certificazione (CA) principale nei componenti personalizzati. Correzioni di bug e miglioramenti • Aggiunge il supporto per i dispositivi Windows che utilizzano una lingua di visualizzazione diversa dall'inglese. • Aggiorna il modo in cui il nucleo Greengrass analizza gli argomenti booleani dell'installatore, in modo da poter specificare un argomento booleano senza un valore booleano per specificare un valore. `true` Ad esempio, ora è possibile specificare invece di installare con il provisioning automatico delle risorse. `--provision --provision true` • Risolve un problema per cui il dispositivo principale non segnalava il proprio stato al servizio AWS IoT Greengrass cloud dopo il provisioning in determinati scenari. • Correzioni e miglioramenti minori aggiuntivi.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.4 il 23 marzo 2022

Questa versione fornisce la versione 2.5.4 del componente Greengrass nucleus e la versione 2.0.10 del componente Lambda launcher.

Data di rilascio: 23 marzo 2022

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.5.4 del Greengrass nucleus. Correzioni di bug e miglioramenti • Correzioni di bug generali e miglioramenti.
Lanciatore Lambda	È disponibile la versione 2.0.10 del componente di avvio Lambda. Correzioni di bug e miglioramenti • Correzioni di bug generali e miglioramenti.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.3 il 6 gennaio 2022

Questa versione fornisce la versione 2.5.3 del componente Greengrass nucleus e il nuovo componente provider PKCS #11.

Data di rilascio: 6 gennaio 2022

Aspetti salienti

- Integrazione della sicurezza hardware: ora puoi configurare il software AWS IoT Greengrass Core per utilizzare una chiave privata e un certificato da archiviare in modo sicuro in un modulo di sicurezza hardware (HSM). Per ulteriori informazioni, consulta Integrazione della sicurezza hardware.

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.5.3 del Greengrass nucleus.

Componente	Dettagli
	Nuove funzionalità • Aggiunge il supporto per l'integrazione della sicurezza hardware. È possibile utilizzare un modulo di sicurezza hardware (HSM) per archiviar e in modo sicuro la chiave privata e il certificato del dispositivo. Per ulteriori informazioni, consulta Integrazione della sicurezza hardware. Correzioni di bug e miglioramenti • Risolve un problema con le eccezioni di runtime mentre il nucleo stabilisce connessioni MQTT con. AWS IoT Core
Fornitore PKCS #11	È disponibile la versione 2.0.0 del componente del provider PKCS #11. Nuove funzionalità • Aggiunge il supporto per l'integrazione della sicurezza hardware. È possibile utilizzare un modulo di sicurezza hardware (HSM) per archiviar e in modo sicuro la chiave privata e il certificato del dispositivo. Per ulteriori informazioni, consulta Integrazione della sicurezza hardware.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.2 il 3 dicembre 2021

Questa versione fornisce la versione 2.5.2 del componente Greengrass nucleus.

Data di rilascio: 3 dicembre 2021

Dettagli di rilascio

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.5.2 del Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema a causa del quale, dopo l'aggiornamento del Greengrass Nucleus, il servizio Windows non si riavvia dopo l'arresto o il riavvio del dispositivo.
AWS IoT Device Defender	È disponibile la versione 3.0.1 del componente. AWS IoT Device Defender Questa versione del AWS IoT Device Defender componente prevede parametri di configurazione diversi rispetto alla versione 2.x. Se si utilizza una configurazione non predefinita per la versione 2.x e si desidera eseguire l'aggiornamento dalla v2.x alla v3.x, è necessario aggiornare la configura zione del componente. Per ulteriori informazioni, consulta Configurazione dei componenti. AWS IoT Device Defender Nuove funzionalità • Aggiunge il supporto per i dispositivi principali che eseguono Windows.

Componente	Dettagli
	• Cambia il tipo di componente da componente Lambda a component e generico. Questo componente ora non dipende più dal componente legacy del router di sottoscrizione per la creazione di abbonamenti. • Aggiunge il nuovo parametro `UseInstaller` di configurazione che consente di disabilitare facoltativamente lo script di installazione che installa le dipendenze dei componenti.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.1 il 23 novembre 2021

Questa versione fornisce la versione 2.5.1 del componente Greengrass nucleus.

Data di rilascio: 23 novembre 2021

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.5.1 del Greengrass nucleus. Correzioni di bug e miglioramenti • Aggiunge il supporto per le versioni a 32 bit di Java Runtime Environment (JRE) su Windows. • Modifica il comportamento di rimozione dei gruppi di oggetti per i dispositivi principali la cui AWS IoT politica non concede l'`greengrass:ListThingGroupsForCoreDevice` autorizzazione. Con questa versione, la distribuzione continua, registra un avviso e non rimuove i componenti quando si rimuove il dispositivo principale da un gruppo di oggetti. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi. • Risolve un problema con le variabili di ambiente di sistema che il nucleo Greengrass mette a disposizione dei processi dei componenti Greengrass. È ora possibile riavviare un componente in modo che utilizzi le variabili di ambiente di sistema più recenti.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.5.0 il 12 novembre 2021

Questa versione fornisce la versione 2.5.0 del componente Greengrass nucleus, AWS nuovi componenti forniti e aggiornamenti ai componenti forniti. AWS

Data di rilascio: 12 novembre 2021

Aspetti salienti

- Supporto per dispositivi Windows: ora puoi eseguire il software AWS IoT Greengrass Core su dispositivi che eseguono sistemi operativi Windows. Per ulteriori informazioni, consultare Piattaforme supportate e requisiti e Compatibilità delle funzionalità Greengrass per sistema operativo.
- Nuovo comportamento di rimozione dei gruppi di cose: ora è possibile rimuovere un dispositivo principale da un gruppo di oggetti per rimuovere i componenti di quel gruppo di oggetti nella successiva distribuzione su quel dispositivo.

> **Important**
>
> A seguito di questa modifica, la AWS IoT policy di un dispositivo principale deve disporre dell'`greengrass:ListThingGroupsForCoreDevice`autorizzazione. Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per effettuare il provisioning delle risorse, lo consente la AWS IoT politica predefinita`greengrass:*`, che include questa autorizzazione. Per ulteriori informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass.

- Supporto per la sicurezza hardware: ora puoi configurare il software AWS IoT Greengrass Core per utilizzare un modulo di sicurezza hardware (HSM), in modo da poter archiviare in modo sicuro la chiave privata e il certificato del dispositivo. Per ulteriori informazioni, consulta Integrazione della sicurezza hardware.
- Supporto proxy HTTPS: ora puoi configurare il software AWS IoT Greengrass Core per la connessione tramite proxy HTTPS. Per ulteriori informazioni, consulta Connessione alla porta 443 o tramite un proxy di rete.

Dettagli sulla versione

- Aggiornamenti del supporto della piattaforma
- aggiornamenti dei componenti pubblici

Aggiornamenti del supporto della piattaforma

Platform (Piattaforma)	Dettagli
Windows	AWS IoT Greengrassora supporta l'esecuzione del software AWS IoT Greengrass Core nelle seguenti versioni di Windows: • Windows 10 • Windows Server 2019 Per ulteriori informazioni, consultare Piattaforme supportate e requisiti e Compatibilità delle funzionalità Greengrass per sistema operativo.

aggiornamenti dei componenti pubblici

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.5.0 del Greengrass nucleus. Nuove funzionalità • Aggiunge il supporto per i dispositivi principali che eseguono Windows. • Modifica il comportamento della rimozione dei gruppi di oggetti. Con questa versione, è possibile rimuovere un dispositivo principale da un gruppo di oggetti per disinstallarne i componenti nella distribuzione successiva. A seguito di questa modifica, la AWS IoT policy di un dispositivo principale deve disporre dell'`greengrass:ListThingGroupsForCoreDevice` autorizzazione. Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per effettuare il provisioning delle risorse, lo consente la AWS IoT politica predefini tag`greengrass:*` , che include questa autorizzazione. Per ulteriori

Componente	Dettagli
	informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass. • Aggiunge il supporto per le configurazioni proxy HTTPS. Per ulteriori informazioni, consulta Connessione alla porta 443 o tramite un proxy di rete. • Aggiunge il nuovo parametro `windowsUser` di configurazione. È possibile utilizzare questo parametro per specificare l'utente predefinito da utilizzare per eseguire i componenti su un dispositivo Windows principale. Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti. • Aggiunge le nuove opzioni di `httpClient` configurazione che è possibile utilizzare per personalizzare i timeout delle richieste HTTP per migliorare le prestazioni su reti lente. Per ulteriori informazioni, vedete il parametro di configurazione HttpClient. Correzioni di bug e miglioramenti • Risolve l'opzione del ciclo di vita di bootstrap per riavviare il dispositivo principale da un componente. • Aggiunge il supporto per i trattini nelle variabili di ricetta. • Corregge l'autorizzazione IPC per i componenti della funzione Lambda su richiesta. • Migliora i messaggi di registro e modifica i log non critici da un `DEBUG` livello `INFO` all'altro, quindi i log sono più utili. • Rimuove l'`iot:DescribeCertificate` autorizzazione dal ruolo di scambio di token predefinito creato dal nucleo Greengrass quando si installa il software AWS IoT Greengrass Core con provisioning automatico. Questa autorizzazione non viene utilizzata dal nucleo Greengrass. • Risolve un problema per cui lo script di provisioning automatico non richiedeva l'`iam:GetPolicy` autorizzazione se `iam:CreatePolicy` disponibile per la stessa politica. • Correzioni e miglioramenti minori aggiuntivi.

Componente	Dettagli
Greengrass CLI	È disponibile la versione 2.5.0 della Greengrass CLI. Nuove funzionalità • Aggiunge il supporto per i dispositivi principali che eseguono Windows. • Aggiunge il nuovo parametro di `AuthorizedWindowsGroups` configurazione che è possibile specificare per autorizzare i gruppi di sistema a utilizzare la Greengrass CLI sui dispositivi Windows. • Aggiunge il `windowsUser` parametro per le distribuzioni locali. È possibile utilizzare questo parametro per specificare l'utente da utilizzare per eseguire i componenti su un dispositivo Windows principale.

Componente	Dettagli
CloudWatch metriche	È disponibile la versione 3.0.0 del componente CloudWatchmetrics. Questa versione del componente CloudWatch metrics prevede parametri di configurazione diversi rispetto alla versione 2.x. Se utilizzi una configurazione non predefinita per la versione 2.x e desideri eseguire l'aggiornamento dalla v2.x alla v3.x, devi aggiornare la configurazione del componente. Per ulteriori informazioni, consulta Configurazione del componente metrics. CloudWatch Nuove funzionalità • Aggiunge il supporto per i dispositivi principali che eseguono Windows. • Cambia il tipo di componente da componente Lambda a component e generico. Questo componente ora non dipende più dal componente legacy del router di sottoscrizione per la creazione di abbonamenti. • Aggiunge un nuovo parametro di `InputTopic` configurazione per specificare l'argomento a cui il componente si iscrive per ricevere messaggi. • Aggiunge un nuovo parametro di `OutputTopic` configurazione per specificare l'argomento in cui il componente pubblica le risposte di stato. • Aggiunge un nuovo parametro di `PubSubToIoTCore` configurazione per specificare se pubblicare e sottoscrivere argomenti AWS IoT Core MQTT. • Aggiunge il nuovo parametro `UseInstaller` di configurazione che consente di disabilitare facoltativamente lo script di installazione che installa le dipendenze dei componenti. Correzioni di bug e miglioramenti Aggiunge il supporto per timestamp duplicati nei dati di input.

Componente	Dettagli
Gestore Lambda	È disponibile la versione 2.2.0 del componente Lambda manager. Correzioni di bug e miglioramenti • Risolve un problema per cui le funzioni Lambda non potevano scrivere i log dopo un riavvio. • Risolve un problema a causa del quale il router di sottoscrizione legacy invia messaggi duplicati quando nell'argomento sono presenti caratteri jolly. • Risolve un problema a causa del quale le funzioni Lambda non bloccate non potevano utilizzare la libreria di comunicazione interprocesso (IPC) Greengrass in. SDK per dispositivi AWS IoT

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.4.0 il 3 agosto 2021

Questa versione fornisce la versione 2.4.0 del componente Greengrass nucleus, AWS nuovi componenti forniti e aggiornamenti ai componenti forniti. AWS

Data di rilascio: 3 agosto 2021

Aspetti salienti

- Limiti delle risorse di sistema: il componente Greengrass nucleus ora supporta i limiti delle risorse di sistema. È possibile configurare la quantità massima di utilizzo di CPU e RAM che i processi di ciascun componente possono utilizzare sul dispositivo principale. Per ulteriori informazioni, consulta Configura i limiti delle risorse di sistema per i componenti.
- Pausa/riprendi componenti: il nucleo di Greengrass ora supporta la pausa e la ripresa dei componenti. È possibile utilizzare la libreria di comunicazione tra processi (IPC) per sviluppare componenti personalizzati che mettono in pausa e riprendono i processi degli altri componenti. Per ulteriori informazioni, consultare PauseComponent e ResumeComponent.
- Installazione con AWS IoT fleet provisioning: utilizza il nuovo plug-in AWS IoT Fleet Provisioning per installare il software AWS IoT Greengrass Core sui dispositivi che si connettono per fornire le risorse necessarie. AWS IoT AWS I dispositivi utilizzano un certificato di attestazione per la fornitura. Puoi incorporare il certificato di richiesta sui dispositivi durante la produzione, in modo che

ciascun dispositivo possa essere fornito non appena viene messo online. Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con il provisioning AWS IoT della flotta.

- Installazione con provisioning personalizzato: sviluppate un plug-in di provisioning personalizzato per fornire AWS le risorse necessarie quando installate il software AWS IoT Greengrass Core sui dispositivi. Puoi creare un'applicazione Java da eseguire durante l'installazione per configurare i dispositivi core Greengrass per il tuo caso d'uso personalizzato. Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning personalizzato delle risorse.

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.4.0 del Greengrass nucleus.

Componente	Dettagli
	Nuove funzionalità • Aggiunge il supporto per i limiti delle risorse di sistema. È possibile configurare la quantità massima di utilizzo di CPU e RAM che i processi di ciascun componente possono utilizzare sul dispositivo principale. Per ulteriori informazioni, consulta Configura i limiti delle risorse di sistema per i componenti. • Aggiunge operazioni IPC per mettere in pausa e riprendere i component i. Per ulteriori informazioni, consultare PauseComponent e ResumeCom ponent. • Aggiunge il supporto per i plugin di provisioning. È possibile specificare un file JAR da eseguire durante l'installazione per fornire AWS le risorse necessarie per un dispositivo core Greengrass. Il nucleo Greengrass include un'interfaccia che è possibile implementare per sviluppare plugin di provisioning personalizzati. Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning personalizzato delle risorse. • Aggiunge l'`thing-name-policy` argomento opzionale al programma di installazione del software CoreAWS IoT Greengrass. È possibile utilizzare questa opzione per specificare una AWS IoT politica esistente o personalizzata quando si installa il software AWS IoT Greengrass Core con il provisioning automatico delle risorse. Correzioni di bug e miglioramenti • Aggiorna la configurazione della registrazione all'avvio. Questo risolve un problema a causa del quale la configurazione di registrazione non veniva applicata all'avvio. • Aggiorna il collegamento simbolico del nucleus loader in modo che punti all'archivio dei componenti nella cartella principale di Greengras s durante l'installazione. Questo aggiornamento consente di eliminare il file JAR e altri artefatti del nucleo scaricati durante l'installazione del software Core. AWS IoT Greengrass • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.

Componente	Dettagli
Greengrass CLI	È disponibile la versione 2.4.0 della Greengrass CLI. Nuove funzionalità • Aggiunge il supporto per i limiti delle risorse di sistema. Quando si crea una distribuzione locale, è possibile configurare la quantità massima di utilizzo di CPU e RAM che i processi di ciascun componente possono utilizzare sul dispositivo principale. Per ulteriori informazioni, consulta Configura i limiti delle risorse di sistema per i componenti e il comando deployment create.
AWS IoTapprovvigionamento della flotta tramite reclamo	Il plug-in AWS IoT Fleet Provisioning by claim è ora disponibile. Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con il provisioning AWS IoT della flotta. Nuove funzionalità • Aggiunge il supporto per l'installazione del software AWS IoT Greengrass Core con il provisioning AWS IoT della flotta. Durante l'installazione, i dispositivi si connettono AWS IoT per fornire AWS le risorse necessarie e e scaricare i certificati dei dispositivi da utilizzare per le normali operazioni.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.3.0 il 29 giugno 2021

Questa versione fornisce la versione 2.3.0 del componente Greengrass nucleus.

Data di rilascio: 29 giugno 2021

Aspetti salienti

- Supporto per configurazioni di grandi dimensioni: il componente Greengrass nucleus ora supporta documenti di distribuzione fino a 10 MB. Ora puoi distribuire aggiornamenti di configurazione più ampi ai componenti Greengrass.

> **Note**
>
> Per utilizzare questa funzionalità, la AWS IoT policy di base del dispositivo deve consentire l'`greengrass:GetDeploymentConfiguration`autorizzazione. Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per effettuare il provisioning delle risorse, lo consente la AWS IoT policy del dispositivo principale`greengrass:*`, che include questa autorizzazione. Per ulteriori informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass.

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.3.0 del Greengrass nucleus.

Componente	Dettagli
	Nuove funzionalità • Aggiunge il supporto per i documenti di configurazione della distribuzione fino a 10 MB, dai 7 KB (per le distribuzioni destinate a oggetti) o 31 KB (per le distribuzioni destinate a gruppi di oggetti). Per utilizzare questa funzionalità, la AWS IoT policy di base del dispositivo deve consentire l'autorizzazione. `greengrass:GetDeploymentConfiguration` Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per effettuare il provisioning delle risorse, lo consente la AWS IoT policy del dispositivo principal `egreengrass:*` , che include questa autorizzazione. Per ulteriori informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass. • Aggiunge la variabile `iot:thingName` recipe. Puoi usare questa variabile di ricetta per ottenere il nome dell'elemento del dispositivo AWS IoT principale in una ricetta. Per ulteriori informazioni, consulta Variabili di ricetta. Correzioni di bug e miglioramenti • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.2.0 il 18 giugno 2021

Questa versione fornisce la versione 2.2.0 del componente Greengrass nucleus, AWS nuovi componenti forniti e aggiornamenti ai componenti forniti. AWS

Data di rilascio: 18 giugno 2021

Aspetti salienti

- Supporto per dispositivi client: i nuovi componenti dei dispositivi AWS client forniti consentono di connettere i dispositivi client ai dispositivi principali utilizzando il cloud discovery. È possibile

sincronizzare i dispositivi client AWS IoT Core e interagire con i dispositivi client nei componenti Greengrass. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

- Servizio shadow locale: il nuovo componente shadow manager abilita il servizio shadow locale sui dispositivi principali. È possibile utilizzare questo servizio shadow per interagire con le ombre locali in modalità offline utilizzando le librerie di comunicazione interprocesso (IPC) Greengrass in. SDK per dispositivi AWS IoT È inoltre possibile utilizzare il componente shadow manager per sincronizzare gli stati shadow locali con. AWS IoT Core Per ulteriori informazioni, consulta Interagisci con le ombre dei dispositivi.

Dettagli sulla versione

- Aggiornamenti pubblici dei componenti

Aggiornamenti pubblici dei componenti

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.2.0 del Greengrass nucleus.

Componente	Dettagli
	Nuove funzionalità • Aggiunge operazioni IPC per la gestione locale delle shadow. Correzioni di bug e miglioramenti • Riduce le dimensioni del file JAR. • Riduce l'utilizzo della memoria. • Risolve i problemi per cui la configurazione del registro non veniva aggiornata in alcuni casi. • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.
Shadow manager	È disponibile la versione 2.0.0 del nuovo componente shadow manager. Nuove funzionalità • Aggiunge il supporto per le ombre classiche e con nome. • Aggiunge il supporto per la gestione locale delle shadow tramite IPC. • Aggiunge il supporto per la sincronizzazione delle shadow con. AWS IoT Core
Autenticazione del dispositivo client	È disponibile la versione 2.0.0 del nuovo componente di autenticazione del dispositivo client. Nuove funzionalità • Aggiunge il supporto per i dispositivi client Greengrass, che sono dispositivi IoT locali che si connettono a un dispositivo principale tramite MQTT. • Aggiunge il supporto per l'autenticazione e l'autorizzazione dei dispositivi client e le relative azioni MQTT.
Broker MQTT Moquette	È disponibile la versione 2.0.0 del nuovo componente del broker Moquette MQTT. Nuove funzionalità • Aggiunge il supporto per un broker Moquette MQTT locale che gestisce la comunicazione con i dispositivi client.

Componente	Dettagli
Bridge MQTT	È disponibile la versione 2.0.0 del nuovo componente bridge MQTT. Nuove funzionalità • Aggiunge il supporto per inoltrare messaggi tra il broker MQTT locale, il broker di pubblicazione/sottoscrizione Greengrass locale e il broker MQTT. AWS IoT Core
Rilevatore IP	È disponibile la versione 2.0.0 del nuovo componente del rilevatore IP. Nuove funzionalità • Aggiunge il supporto per segnalare gli endpoint del broker MQTT locale di un dispositivo principale al servizio AWS IoT Greengrass cloud per consentire la connessione dei dispositivi client.
Gestore dei registri	È disponibile la versione 2.1.1 del componente log manager. Correzioni di bug e miglioramenti • Risolve un problema per cui la configurazione del registro di sistema non veniva aggiornata in alcuni casi.
Rilevamento di oggetti DLR	È disponibile la versione 2.1.2 del rilevamento di oggetti DLR. Correzioni di bug e miglioramenti • Risolve un problema di ridimensionamento dell'immagine che causava riquadri di delimitazione imprecisi nei risultati di inferenza del rilevamen to di oggetti DLR di esempio.
TensorFlow Rilevamento di oggetti Lite	È disponibile la versione 2.1.1 del rilevamento di oggetti TensorFlow Lite. Correzioni di bug e miglioramenti • Risolve un problema di ridimensionamento dell'immagine che causava riquadri di delimitazione imprecisi nei risultati di inferenza del rilevamen to di oggetti TensorFlow Lite di esempio.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.1.0 il 26 aprile 2021

Questa versione fornisce la versione 2.1.0 del componente Greengrass nucleus e AWS aggiorna i componenti forniti.

Data di rilascio: 26 aprile 2021

Aspetti salienti

- Integrazione tra Docker Hub e Amazon Elastic Container Registry (Amazon ECR): il nuovo componente Docker Application Manager consente di scaricare immagini pubbliche o private da Amazon ECR. Puoi utilizzare questo componente anche per scaricare immagini pubbliche da Docker Hub e. Marketplace AWS Per ulteriori informazioni, consulta Esegui un contenitore Docker.
- Immagini Dockerfile e Docker per il software AWS IoT Greengrass Core: puoi usare l'immagine Docker Greengrass per eseguirla in AWS IoT Greengrass un contenitore Docker che utilizza Amazon Linux 2 come sistema operativo di base. Puoi anche usare il AWS IoT Greengrass Dockerfile per creare la tua immagine Greengrass. Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core in un contenitore Docker.
- Supporto per framework e piattaforme di machine learning aggiuntivi: è possibile distribuire componenti di inferenza di machine learning di esempio che utilizzano modelli pre-addestrati per eseguire la classificazione delle immagini di esempio e il rilevamento degli oggetti utilizzando TensorFlow Lite 2.5.0 e DLR 1.6.0. Questa versione estende anche il supporto di machine learning di esempio per i dispositivi Armv8 (AArch64). Per ulteriori informazioni, consulta Esecuzione dell'inferenza di Machine Learning.

Dettagli sulla versione

- Aggiornamenti del supporto della piattaforma
- Aggiornamenti dei componenti pubblici

Aggiornamenti del supporto della piattaforma

Platform (Piattaforma)	Dettagli
Docker	AWS IoT GreengrassSono ora disponibili un Dockerfile e un'immagine Docker per. Dockerfile AWS IoT Greengrassfornisce un Dockerfile per creare un'immagine del contenitore con software AWS IoT Greengrass Core e dipendenze installati su un'immagine base Amazon Linux 2 (x86_64). Puoi modificar e l'immagine di base nel Dockerfile per eseguirla su un'architettura di piattaforma diversa. AWS IoT Greengrass immagine Docker AWS IoT Greengrassfornisce un'immagine Docker predefinita con software e dipendenze AWS IoT Greengrass Core installati su un'immagi ne base Amazon Linux 2 (x86_64). Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core in un contenitore Docker.

Aggiornamenti dei componenti pubblici

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi.

Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli

Nucleo Greengrass	È disponibile la versione 2.1.0 del Greengrass nucleus.
	
	Nuove funzionalità
	• Supporta il download di immagini Docker da archivi privati in Amazon ECR.
	• Aggiunge i seguenti parametri per personalizzare la configurazione MQTT sui dispositivi principali:
	  • `maxInFlightPublishes` — Il numero massimo di messaggi MQTT QoS 1 non riconosciuti che possono essere in transito contemporaneamente.
	  • `maxPublishRetry` — Il numero massimo di volte in cui riprovare un messaggio che non viene pubblicato.
	• Aggiunge il parametro di `fleetstatusservice` configurazione per configurare l'intervallo con cui il dispositivo principale pubblica lo stato del dispositivo su. Cloud AWS
	• Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.
	
	Correzioni di bug e miglioramenti
	• Risolve un problema che causava la duplicazione delle distribuzioni shadow al riavvio del nucleo.
	• Risolve un problema che causava l'arresto anomalo del nucleus quando rilevava un'eccezione di carico del servizio.
	• Migliora la risoluzione delle dipendenze dei componenti per evitare il fallimento di una distribuzione che include una dipendenza circolare.

Componente	Dettagli
	• Risolve un problema che impediva la ridistribuzione di un componente del plug-in se tale componente era stato precedentemente rimosso dal dispositivo principale. • Risolve un problema che causava l'impostazione della variabile di `HOME` ambiente nella */greengrass/v2* `/work` directory per i componenti Lambda o per i componenti eseguiti come root. La `HOME` variabile è ora impostata correttamente nella home directory dell'utente che esegue il componente. • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.
Gestore di applicazioni Docker	È disponibile la versione 2.0.0 del nuovo componente Docker Application Manager. Nuove funzionalità • Gestisce le credenziali per scaricare immagini da archivi privati in Amazon ECR. • Scarica immagini pubbliche da Amazon ECR, Docker Hub e. Marketplace AWS
Lanciatore Lambda	È disponibile la versione 2.0.4 del componente di avvio Lambda. Correzioni di bug e miglioramenti • Risolve un problema a causa del quale il componente non passa correttamente `AddGroupOwner` al contenitore delle funzioni Lambda.
Router di abbonamento legacy	È disponibile la versione 2.1.0 del componente legacy del router di abbonamento. Correzioni di bug e miglioramenti • Aggiunge il supporto per specificare i nomi dei componenti anziché gli ARN per `source` e. `target` Se si specifica il nome di un componente per un abbonamento, non è necessario riconfigurare l'abbonamento ogni volta che cambia la versione della funzione Lambda.

Componente	Dettagli
Console di debug locale	È disponibile la versione 2.1.0 del componente della console di debug locale. Nuove funzionalità • Utilizza HTTPS per proteggere la connessione alla console di debug locale. HTTPS è abilitato per impostazione predefinita. Correzioni di bug e miglioramenti • Puoi eliminare i messaggi della flashbar nell'editor di configurazione.
Gestore dei registri	È disponibile la versione 2.1.0 del componente log manager. Correzioni di bug e miglioramenti • Usa i valori predefiniti per `logFileDirectoryPath` e `logFileRegex` che funzionano per i componenti Greengrass che stampano su output standard (stdout) ed errore standard (stderr). • Indirizza correttamente il traffico attraverso un proxy di rete configurato durante il caricamento dei log su Logs. CloudWatch • Gestisci correttamente i due punti (`:`) nei nomi dei flussi di log. CloudWatch I nomi dei flussi di log non supportano i due punti. • Semplifica i nomi dei flussi di log rimuovendo i nomi dei gruppi di oggetti dal flusso di log. • Rimuove un messaggio di registro degli errori che viene stampato durante il normale comportamento.

Componente	Dettagli
Classificazione delle immagini DLR	È disponibile la versione 2.1.1 del componente di classificazione delle immagini DLR. Nuove funzionalità • Usa Deep Learning Runtime v1.6.0. • Aggiungi il supporto per la classificazione delle immagini di esempio sulle piattaforme Armv8 (AArch64). Ciò estende il supporto dell'appr endimento automatico per i dispositivi core Greengrass che eseguono NVIDIA Jetson, come Jetson Nano. • Abilita l'integrazione della fotocamera per l'inferenza dei campioni. Utilizzate il nuovo parametro di `UseCamera` configurazione per consentire al codice di inferenza di esempio di accedere alla telecamera sul dispositivo principale Greengrass ed eseguire l'inferenza localmente sull'immagine acquisita. • Aggiungi il supporto per la pubblicazione dei risultati di inferenza a. Cloud AWS Utilizzate il nuovo parametro di `PublishResultsOnTopic` configurazione per specificare l'argomento su cui desiderate pubblicare i risultati. • Aggiungete il nuovo parametro di `ImageDirectory` configurazione che consente di specificare una directory personalizzata per l'immagine su cui desiderate eseguire l'inferenza. Correzioni di bug e miglioramenti • Scrivi i risultati dell'inferenza nel file di registro del componente anziché in un file di inferenza separato. • Utilizzate il modulo di registrazione del software AWS IoT Greengrass Core per registrare l'output dei componenti. • Utilizzate il SDK per dispositivi AWS IoT per leggere la configurazione del componente e applicare le modifiche alla configurazione.

Componente	Dettagli
Rilevamento di oggetti DLR	È disponibile la versione 2.1.1 del componente DLR per il rilevamento di oggetti. Nuove funzionalità • Usa Deep Learning Runtime v1.6.0. • Aggiungi il supporto per il rilevamento di oggetti di esempio sulle piattaforme Armv8 (AArch64). Ciò estende il supporto dell'appr endimento automatico per i dispositivi core Greengrass che eseguono NVIDIA Jetson, come Jetson Nano. • Abilita l'integrazione della fotocamera per l'inferenza dei campioni. Utilizzate il nuovo parametro di `UseCamera` configurazione per consentire al codice di inferenza di esempio di accedere alla telecamera sul dispositivo principale Greengrass ed eseguire l'inferenza localmente sull'immagine acquisita. • Aggiungi il supporto per la pubblicazione dei risultati di inferenza a. Cloud AWS Utilizzate il nuovo parametro di `PublishResultsOnTo pic` configurazione per specificare l'argomento su cui desiderate pubblicare i risultati. • Aggiungete il nuovo parametro di `ImageDirectory` configurazione che consente di specificare una directory personalizzata per l'immagine su cui desiderate eseguire l'inferenza. Correzioni di bug e miglioramenti • Scrivi i risultati dell'inferenza nel file di registro del componente anziché in un file di inferenza separato. • Utilizzate il modulo di registrazione del software AWS IoT Greengrass Core per registrare l'output dei componenti. • Utilizzate il SDK per dispositivi AWS IoT per leggere la configurazione del componente e applicare le modifiche alla configurazione.

Componente	Dettagli
Archivio dei modelli di classificazione delle immagini DLR	È disponibile la versione 2.1.1 del componente DLR Image Classification Model Store. Nuove funzionalità • Aggiungere un modello di classificazione delle immagini ResNet -50 di esempio per le piattaforme Armv8 (AArch64). Ciò estende il supporto dell'apprendimento automatico per i dispositivi core Greengrass che eseguono NVIDIA Jetson, come Jetson Nano.
Archivio di modelli di rilevamento di oggetti DLR	È disponibile la versione 2.1.1 del componente DLR Object Detection Model Store. Nuove funzionalità • Aggiungi un modello di rilevamento di oggetti YOLov3 di esempio per piattaforme Armv8 (AArch64). Ciò estende il supporto dell'appr endimento automatico per i dispositivi core Greengrass che eseguono NVIDIA Jetson, come Jetson Nano.
Programma di installazione DLR	È disponibile la versione 1.6.1 del componente DLR. Nuove funzionalità • Installa Deep Learning Runtime v1.6.0 e le sue dipendenze. • Aggiungi il supporto per l'installazione di DLR su piattaforme Armv8 (AArch64). Ciò estende il supporto dell'apprendimento automatico per i dispositivi core Greengrass che eseguono NVIDIA Jetson, come Jetson Nano. Correzioni di bug e miglioramenti • Installa il SDK per dispositivi AWS IoT nell'ambiente virtuale per leggere la configurazione del componente e applicare le modifiche alla configura zione. • Correzioni e miglioramenti aggiuntivi di bug minori.

Componente	Dettagli
TensorFlow Classificazione delle immagini Lite	È disponibile la versione 2.1.0 del nuovo componente di classificazione delle immagini TensorFlow Lite. Nuove funzionalità • Aggiunge il supporto per l'inferenza della classificazione delle immagini di esempio utilizzando TensorFlow Lite.
TensorFlow Rilevamento di oggetti Lite	È disponibile la versione 2.1.0 del nuovo componente TensorFlow Lite per il rilevamento di oggetti. Nuove funzionalità • Aggiunge il supporto per l'inferenza del rilevamento di oggetti di esempio utilizzando TensorFlow Lite.
TensorFlo w Archivio di modelli di classificazione delle immagini Lite	È disponibile la versione 2.1.0 del nuovo componente TensorFlow Lite Image Classification Model Store. Nuove funzionalità • Fornisci un modello quantizzato MobileNet v1 pre-addestrato per l'inferenza della classificazione delle immagini di esempio utilizzando Lite. TensorFlow
TensorFlow Archivio di modelli Lite per il rilevamento di oggetti	È disponibile la versione 2.1.0 del nuovo componente TensorFlow Lite object detection model store. Nuove funzionalità • Fornisci un MobileNet modello Single Shot Detection (SSD) preaddest rato sul set di dati COCO per l'inferenza del rilevamento di oggetti campione utilizzando Lite. TensorFlow
TensorFlow Lite	È disponibile la versione 2.5.0 del nuovo componente TensorFlow Lite. Nuove funzionalità • Installa TensorFlow Lite v1.6.0 e le sue dipendenze in un ambiente virtuale su piattaforme Armv7, Armv8 (AArch64) e x86_64.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.0.5 il 09 marzo 2021

Questa versione fornisce la versione 2.0.5 del componente Greengrass nucleus e AWS aggiorna i componenti forniti. Risolve un problema relativo al supporto proxy di rete e un problema con l'endpoint AWS del piano dati Greengrass nelle regioni cinesi.

Data di rilascio: 09 marzo 2021

Aggiornamenti dei componenti pubblici

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.0.5 del Greengrass nucleus. Correzioni di bug e miglioramenti • Instrada correttamente il traffico attraverso un proxy di rete configurato durante il download dei AWS componenti forniti.

Componente	Dettagli
	• Usa l'endpoint del piano dati Greengrass corretto nelle AWS regioni cinesi.

Rilascio: aggiornamento del software AWS IoT Greengrass Core v2.0.4 il 04 febbraio 2021

Questa versione fornisce la versione 2.0.4 del componente Greengrass nucleus. Include il nuovo `greengrassDataPlanePort` parametro per configurare la comunicazione HTTPS sulla porta 443 e corregge i bug. La policy IAM minima ora richiede l'indicazione `iam:GetPolicy` e il momento in `sts:GetCallerIdentity` cui viene eseguito il programma di installazione del software AWS IoT Greengrass Core. `--provision true`

Data di rilascio: 04 febbraio 2021

Aggiornamenti dei componenti pubblici

La tabella seguente elenca AWS i componenti forniti che includono funzionalità nuove e aggiornate.

> **Important**
>
> Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Componente	Dettagli
Nucleo Greengrass	È disponibile la versione 2.0.4 del Greengrass nucleus. Nuove funzionalità • Abilita il traffico HTTPS sulla porta 443. È possibile utilizzare il nuovo parametro di `greengrassDataPlanePort` configurazione per la versione 2.0.4 del componente nucleus per configurare la comunicaz ione HTTPS in modo che viaggi sulla porta 443 anziché sulla porta predefinita 8443. Per ulteriori informazioni, consulta Configura HTTPS sulla porta 443. • Aggiunge la variabile di ricetta del percorso di lavoro. È possibile utilizzare questa variabile di ricetta per ottenere il percorso delle cartelle di lavoro dei componenti, che è possibile utilizzare per condividere file tra i componenti e le relative dipendenze. Per ulteriori informazioni, consulta la variabile di ricetta del percorso di lavoro. Correzioni di bug e miglioramenti • Impedisce la creazione della politica di ruolo dello scambio di token AWS Identity and Access Management (IAM) se esiste già una politica di ruolo. Come risultato di questa modifica, il programma di installazione ora richiede `iam:GetPolicy` e `sts:GetCallerIdentity` quando viene eseguito con`--provision true` . Per ulteriori informazioni, consulta Policy IAM minima per l'installatore per il provisioning delle risorse. • Gestisce correttamente l'annullamento di una distribuzione che non è stata ancora registrata correttamente. • Aggiorna la configurazione per rimuovere le voci più vecchie con timestamp più recenti durante il rollback di una distribuzione. • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.

Migrazione dalla AWS IoT Greengrass versione 1

AWS IoT Greengrass Version 2è una versione principale del software AWS IoT Greengrass Core, delle API e della console. AWS IoT Greengrass V2introduce diversi miglioramentiAWS IoT Greengrass V1, come applicazioni modulari, implementazioni su grandi flotte di dispositivi e supporto per piattaforme aggiuntive.

> **Note**
>
> Dopo il 30 giugno 2023 AWS IoT Greengrass Version 1 non riceve più aggiornamenti di funzionalità, miglioramenti, correzioni di bug o patch di sicurezza. Per ulteriori informazioni, consulta la politica di manutenzione. AWS IoT Greengrass V1 Se lo utilizziAWS IoT Greengrass V1, ti consigliamo vivamente di migrare aAWS IoT Greengrass V2.

Segui le istruzioni di questa guida per migrare da AWS IoT Greengrass V1 a. AWS IoT Greengrass V2

Posso eseguire le mie applicazioni V1 sulla V2?

La maggior parte delle applicazioni V1 può essere eseguita su dispositivi core V2 senza dover modificare il codice dell'applicazione. Se le tue applicazioni V1 utilizzano la seguente funzionalità, non sarai in grado di eseguirle su V2.

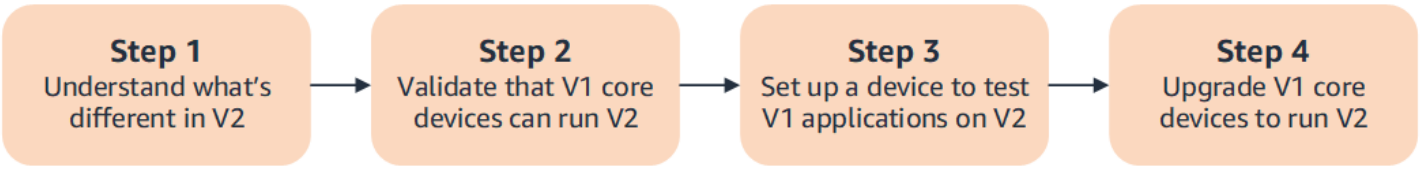
- I runtime delle funzioni Lambda C e C++

Se le applicazioni V1 utilizzano una delle seguenti funzionalità, è necessario modificare il codice dell'applicazione per utilizzare la SDK per dispositivi AWS IoT V2 su cui eseguire le applicazioni. AWS IoT Greengrass V2

- Interagisci con il servizio shadow locale
- Pubblica messaggi su dispositivi connessi locali (dispositivi Greengrass)

Panoramica della migrazione

Ad alto livello, è possibile utilizzare la seguente procedura per aggiornare i dispositivi principali da AWS IoT Greengrass V1 aAWS IoT Greengrass V2. La procedura esatta da seguire dipende dai requisiti specifici dell'ambiente in uso.



1. Comprendi le differenze tra V1 e V2

   AWS IoT Greengrass V2introduce nuovi concetti fondamentali per le flotte di dispositivi e il software implementabile, mentre la V2 semplifica diversi concetti della V1.

   Il servizio AWS IoT Greengrass V2 cloud e il software AWS IoT Greengrass Core v2.x non sono retrocompatibili con il servizio cloud e il software Core v1.x. AWS IoT Greengrass V1 AWS IoT Greengrass Di conseguenza, gli aggiornamenti AWS IoT Greengrass V1 over-the-air (OTA) non possono aggiornare i dispositivi principali dalla V1 alla V2.

2. Verifica che i dispositivi principali V1 possano eseguire la V2

   Verifica che un dispositivo core V1 sia in grado di eseguire il software Core v2.x e le funzionalitàAWS IoT Greengrass. AWS IoT Greengrass V2 AWS IoT Greengrass V2ha requisiti di dispositivo diversi da. AWS IoT Greengrass V1

3. Configura un nuovo dispositivo per testare le applicazioni V1 su V2

   Per ridurre al minimo i rischi per i dispositivi in produzione, crea un nuovo dispositivo per testare le applicazioni V1 sulla V2. Dopo aver installato il software AWS IoT Greengrass Core v2.x, puoi creare e distribuire AWS IoT Greengrass V2 componenti per migrare e testare le tue applicazioni. AWS IoT Greengrass V1

4. Aggiorna i dispositivi principali V1 per eseguire la versione 2

   Aggiorna un dispositivo principale V1 esistente per eseguire il software AWS IoT Greengrass Core v2.x e i componenti. AWS IoT Greengrass V2 Per migrare una flotta di dispositivi dalla V1 alla V2, ripeti questo passaggio per ogni dispositivo del parco dispositivi.

Differenze tra AWS IoT Greengrass V1 e AWS IoT Greengrass V2

AWS IoT Greengrass V2 introduce nuovi concetti fondamentali per dispositivi, flotte e software implementabile. Questa sezione descrive i concetti della V1 che sono diversi nella V2.

Concetti e terminologia di Greengrass

Concetto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Codice dell'applicazione	Nel AWS IoT Greengras s V1, le funzioni Lambda definiscono il software che viene eseguito sui dispositi vi principali. In ogni gruppo Greengrass, definisci gli abbonamenti e le risorse locali utilizzate dalla funzione. Per le funzioni Lambda eseguite dal software AWS IoT Greengrass Core in un ambiente di runtime Lambda containerizzato, è possibile definire i parametri del contenitore, come i limiti di memoria.	In AWS IoT Greengrass V2, i componenti sono i moduli software che vengono eseguiti sui dispositivi principali. • Ogni componente ha una ricetta che definisce i metadati, i parametri, le dipendenze e gli script del componente da eseguire in ogni fase del ciclo di vita del componente. • La ricetta definisce anche gli artefatti del component e, che sono file binari, come script, codice compilato e risorse statiche. • Quando si distribuisce un componente su un dispositi vo principale, il dispositi vo principale scarica la ricetta del componente e gli artefatti per eseguire il componente. È possibile importare le funzioni Lambda V1 come componenti eseguiti in un

Concetto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		ambiente di runtime Lambda in. AWS IoT Greengrass V2 Quando si importa la funzione Lambda, si specifica no gli abbonamenti, le risorse locali e i parametri del contenitore per la funzione. Per ulteriori informazioni, consulta Fase 2: Creare e distribuire componenti per migrare le applicazioni AWS IoT Greengrass V2AWS IoT Greengrass V1. Per ulteriori informazioni su come creare component i personalizzati, consulta. Sviluppa AWS IoT Greengrass componenti

Concetto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
AWS IoT Greengrass gruppi e distribuzioni	In AWS IoT Greengrass V1, un gruppo definisce il dispositivo principale, le impostazioni e il software per quel dispositivo principale e l'elenco di AWS IoT elementi che possono connettersi a quel dispositivo principale. Si crea una distribuzione per inviare la configurazione di un gruppo a un dispositivo principale.	In AWS IoT Greengrass V2, si utilizzano le distribuzioni per definire i componenti e le configurazioni software da eseguire sui dispositivi principali. • Ogni distribuzione si rivolge a un singolo dispositivo core (che è una AWS IoT cosa) o a un AWS IoT gruppo di oggetti che può contenere più dispositivi core. • Le distribuzioni ai gruppi di oggetti sono continue, quindi quando si aggiunge un dispositivo principale a un gruppo di oggetti, questo riceve la configurazione software per quel gruppo. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi. In AWS IoT Greengrass V2, puoi anche creare distribuzioni locali utilizzando la CLI di Greengrass per testare componenti software personalizzati sul dispositivo su cui li sviluppi. Per ulteriori informazi

Concetto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		oni, consulta Crea AWS IoT Greengrass componenti.
AWS IoT Greengrass Software di base	Nel AWS IoT Greengras s V1, il software AWS IoT Greengrass Core è un unico pacchetto che contiene il software e tutte le sue funzionalità. Il dispositivo edge su cui si installa il software AWS IoT Greengrass Core è chiamato core Greengrass.	In AWS IoT Greengrass V2, il software AWS IoT Greengras s Core è modulare, quindi puoi scegliere cosa installare per controllare l'ingombro della memoria. • Il componente Greengras s nucleus è l'installazione minima richiesta del AWS IoT Greengrass software Core. Il dispositivo edge su cui si installa il nucleo è chiamato dispositivo core Greengrass. • Il nucleo gestisce le implementazioni, l'orchest razione e la gestione del ciclo di vita di altri componenti sul dispositivo principale. • Funzionalità come stream manager, secret manager e log manager sono componenti che puoi distribuire solo quando ne hai bisogno. Per ulteriori informazioni, consulta AWS-componenti forniti.

Concetto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Connectors (Connettori)	In AWS IoT Greengrass V1, i connettori sono moduli predefiniti che puoi distribui re sui dispositivi AWS IoT Greengrass V1 principali per interagire con l'infrastruttura locale, i protocolli dei dispositi vi e altri AWS servizi cloud.	In AWS IoT Greengrass V2, AWS fornisce componenti Greengrass che implement ano le funzionalità fornite dai connettori in V1. I seguenti AWS IoT Greengrass V2 componenti forniscono la funzionalità del connettore Greengrass V1: • CloudWatch componente metrico • AWS IoT Device Defender componente • Componente Firehose • Componente dell'adattatore di protocollo Modbus-RTU • Componente Amazon SNS Per ulteriori informazioni, consulta AWS-componenti forniti.

Concetto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Dispositivi collegati (dispositivi Greengrass)	Nel AWS IoT Greengras s V1, i dispositivi connessi sono AWS IoT elementi che si aggiungono a un gruppo Greengrass per connettersi al dispositivo principale di quel gruppo e comunicare tramite MQTT. È necessario distribui re quel gruppo ogni volta che si aggiunge o si rimuove un dispositivo connesso. Gli abbonamenti vengono utilizzat i per inoltrare messaggi tra dispositivi connessi e applicazi oni sul dispositivo principale. AWS IoT Core	Nel AWS IoT Greengrass V2, i dispositivi collegati sono chiamati dispositivi client Greengrass. • Associate i dispositivi client ai dispositivi principali per collegarli e comunicare tramite MQTT. • Per autorizzare i dispositi vi client a connettersi, si definiscono politiche di autorizzazione che possono essere applicate a gruppi di dispositivi client, quindi non è necessario creare una distribuzione per aggiunger e o rimuovere un dispositivo client. • Per inoltrare messaggi tra dispositivi client e componenti Greengrass AWS IoT Core, è possibile configurare un componente bridge MQTT opzionale. In entrambi AWS IoT Greengrass V1 i casi AWS IoT Greengrass V2, i dispositivi possono eseguire FreerTOS o utilizzare l'API di scoperta SDK per dispositivi AWS IoTo Greengrass per ottenere informazioni sui dispositi

Concetto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		vi principali a cui possono connettersi. L'API Greengrass discovery è retrocompatibile, quindi se disponi di dispositivi client che si connettono a un dispositivo core V1, puoi collegarli a un dispositivo core V2 senza modificarne il codice. Per ulteriori informazioni sui dispositivi client, vedere. Interagisci con dispositivi IoT locali

Concetto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Risorse locali	In AWS IoT Greengrass V1, le funzioni Lambda eseguite nei contenitori possono essere configurate per accedere a volumi e dispositivi sul file system del dispositivo principale. Queste risorse del file system sono note come risorse locali.	In AWS IoT Greengrass V2, puoi eseguire componenti che sono funzioni Lambda, contenitori Docker o processi nativi del sistema operativo o runtime personalizzati. • Quando si importa una funzione Lambda containerizzata come componente, è necessario specificare le risorse locali utilizzate dalla funzione. • Le funzioni Lambda non containerizzate e i componenti non Lambda possono funzionare direttamente con le risorse locali sui dispositivi principali, quindi non è necessario specificare le risorse locali utilizzate dal componente.

Concetto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Servizio shadow locale	In AWS IoT Greengrass V1, il servizio shadow locale è abilitato per impostazione predefinita e supporta solo ombre classiche senza nome. Utilizzi AWS IoT Greengras s Core SDK nelle funzioni Lambda per interagire con le ombre sui tuoi dispositivi.	In AWS IoT Greengrass V2, abiliti il servizio shadow locale distribuendo il componente shadow manager. • Puoi utilizzare la versione SDK per dispositivi AWS IoT 2 nelle funzioni Lambda e nei componenti personali zzati per interagire con le ombre sui tuoi dispositivi. • Il servizio shadow locale supporta le ombre denominate. • Il servizio shadow locale consente di eliminare le ombre e sincronizzare le ombre eliminate con. AWS IoT Core Per ulteriori informazioni, consulta Interagisci con le ombre dei dispositivi.

Concetto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
Sottoscrizioni	Nel AWS IoT Greengrass V1, si definiscono gli abbonamenti per un gruppo Greengrass per specificare i canali di comunicazione tra le funzioni Lambda, i connettori, i dispositivi collegati, il broker MQTT e AWS IoT Core il servizio shadow locale. Gli abbonamenti specificano dove le funzioni Lambda ricevono messaggi di eventi da utilizzare come payload di funzioni.	In AWS IoT Greengrass V2, si specificano i canali di comunicazione senza utilizzare abbonamenti. • I componenti gestiscono i propri canali di comunicazione per interagire con i messaggi locali di pubblicazione/sottoscrizione, i messaggi AWS IoT Core MQTT e il servizio shadow locale. • Per sviluppare un componente che reagisca ai messaggi provenienti da un altro componente o dal broker AWS IoT Core MQTT, è possibile utilizzare interfacce di comunicazione tra processi (IPC) per la messaggistica locale di pubblicazione/sottoscrizione e la messaggistica MQTT. AWS IoT Core • Per sviluppare un componente che interagisca con il servizio shadow locale, è possibile utilizzare l'interfaccia IPC per il servizio shadow locale. • Nella configurazione del componente, si definisco

Concetto	AWS IoT Greengrass V1	AWS IoT Greengrass V2
		no le politiche di autorizza zione per specificare gli argomenti e le shadow locali che il componente è autorizzato a utilizzare. • Per configurare i canali di comunicazione tra i dispositivi client, il broker di pubblicazione/sottoscrizion e locale e il broker AWS IoT Core MQTT, è necessario configurare e distribuire il componente bridge MQTT. Il componente bridge MQTT consente di interagire con i dispositivi client all'interno dei componenti e di inoltrare messaggi tra dispositivi client e. AWS IoT Core
Accedere ad altri Servizi AWS	In AWS IoT Greengrass V1, si assegna un ruolo AWS Identity and Access Management (IAM), chiamato ruolo di gruppo, a un gruppo Greengrass. Il ruolo di gruppo definisce le autorizzazioni che le funzioni AWS IoT Greengrass e le funzional ità Lambda sul dispositivo principale di quel gruppo utilizzano per accedere. Servizi AWS	In AWS IoT Greengrass V2, si collega un alias di AWS IoT ruolo a un dispositi vo principale Greengrass. L'alias del ruolo rimanda a un ruolo IAM chiamato token exchange role. Il ruolo di scambio di token definisce le autorizzazioni utilizzate dai componenti Greengrass sul dispositivo principale per accedere. Servizi AWS Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

Convalida che i dispositivi core V1 possono eseguire il software V2

Il softwareAWS IoT Greengrass Core v2.x ha requisiti diversi rispetto al softwareAWS IoT Greengrass Core v1.x. Prima di aggiornare i dispositivi core V1 alla V2, consulta i requisiti del dispositivo perAWS IoT Greengrass V2. AWS IoT Greengrass V2attualmente non supporta la migrazione per sistemi personalizzati basati su Linux che utilizzano il progetto Yocto.

È possibile utilizzare AWS IoT Device Tester(IDT) perAWS IoT Greengrass V2 verificare che i dispositivi soddisfino i requisiti per eseguire il softwareAWS IoT Greengrass Core v2.x. IDT è un framework di test scaricabile che viene eseguito sul computer host e si collega ai dispositivi da convalidare. Segui le istruzioni per utilizzare IDT per eseguire la suite diAWS IoT Greengrass qualificazione. Quando configuri IDT, puoi scegliere di convalidare se i dispositivi supportano funzionalità opzionali, come Docker, apprendimento automatico (ML), gestione del flusso di dati e integrazione della sicurezza hardware.

Se IDT segnala errori o errori nei test V2 per un dispositivo core V1, non puoi aggiornare quel dispositivo dalla V1 alla V2.

Configura un nuovo dispositivo core V2 per testare le applicazioni V1

Configura un nuovo dispositivo AWS IoT Greengrass V2 principale per distribuire e testare i componenti e AWS Lambda le funzioni AWS forniti per le tue applicazioni. AWS IoT Greengrass V1 Puoi anche utilizzare questo dispositivo core V2 per sviluppare e testare componenti Greengrass personalizzati aggiuntivi che eseguono processi nativi sui dispositivi principali. Dopo aver testato le applicazioni su un dispositivo core V2, è possibile aggiornare i dispositivi core V1 esistenti alla V2 e distribuire i componenti V2 che forniscono le funzionalità V1.

Fase 1: Installazione su un nuovo dispositivo AWS IoT Greengrass V2

Installa il software AWS IoT Greengrass Core v2.x su un nuovo dispositivo. Puoi seguire il tutorial introduttivo per configurare un dispositivo e imparare a sviluppare e distribuire componenti. Questo tutorial utilizza il provisioning automatico per configurare rapidamente un dispositivo. Quando installi il software AWS IoT Greengrass Core v2.x, specifica l'`--deploy-dev-tools`argomento per distribuire la Greengrass CLI, in modo da poter sviluppare, testare ed eseguire il debug dei componenti direttamente sul dispositivo. Per ulteriori informazioni su altre opzioni di installazione,

incluso come installare il software AWS IoT Greengrass Core dietro un proxy o utilizzare un modulo di sicurezza hardware (HSM), consulta. Installare il software AWS IoT Greengrass Core.

(Facoltativo) Abilita la registrazione su Amazon CloudWatch Logs

Per consentire a un dispositivo core V2 di caricare i log su Amazon CloudWatch Logs, puoi distribuire il componente di gestione dei AWS log fornito. Puoi utilizzare CloudWatch Logs per visualizzare i log dei componenti, in modo da eseguire il debug e la risoluzione dei problemi senza accedere al file system del dispositivo principale. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Fase 2: Creare e distribuire componenti per migrare le applicazioni AWS IoT Greengrass V2AWS IoT Greengrass V1

È possibile eseguire la maggior parte delle applicazioni su. AWS IoT Greengrass V1 AWS IoT Greengrass V2 È possibile importare funzioni Lambda come componenti eseguibili AWS IoT Greengrass V2 e utilizzare componenti AWS forniti che offrono le stesse funzionalità dei connettori. AWS IoT Greengrass

Puoi anche sviluppare componenti personalizzati per creare qualsiasi funzionalità o runtime da eseguire sui dispositivi core Greengrass. Per informazioni su come sviluppare e testare i componenti localmente, vedereCrea AWS IoT Greengrass componenti.

Argomenti

- Importa funzioni Lambda V1
- Usa connettori V1
- Esegui contenitori Docker
- Esegui l'inferenza dell'apprendimento automatico
- Dispositivi Connect V1 Greengrass
- Abilita il servizio shadow locale
- Integrazione con AWS IoT SiteWise

Importa funzioni Lambda V1

È possibile importare funzioni Lambda come AWS IoT Greengrass V2 componenti. Scegli tra i seguenti approcci:

- Importa le funzioni Lambda V1 direttamente come componenti Greengrass.
- Aggiorna le funzioni Lambda per utilizzare le librerie Greengrass nella SDK per dispositivi AWS IoT v2, quindi importa le funzioni Lambda come componenti Greengrass.
- Crea componenti personalizzati che utilizzano codice non Lambda e SDK per dispositivi AWS IoT v2 per implementare le stesse funzionalità delle tue funzioni Lambda.

Se la tua funzione Lambda utilizza funzionalità, come stream manager o local secret, devi definire le dipendenze dai componenti AWS forniti che racchiudono queste funzionalità. Quando si distribuisce il componente della funzione Lambda, la distribuzione include anche il componente per ogni funzionalità definita come dipendenza. Durante la distribuzione, puoi configurare parametri, ad esempio quali segreti distribuire sul dispositivo principale. Non tutte le funzionalità della V1 richiedono una dipendenza dai componenti per la funzione Lambda sulla V2. L'elenco seguente descrive come utilizzare le funzionalità V1 nel componente della funzione Lambda V2.

- Accedere ad altri servizi AWS

  Se la funzione Lambda utilizza AWS credenziali per effettuare richieste ad altri AWS servizi, il ruolo di scambio di token del dispositivo principale deve consentire al dispositivo principale di eseguire AWS le operazioni utilizzate dalla funzione Lambda. Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

- Gestore di flussi

  Se la tua funzione Lambda utilizza stream manager, specifica `aws.greengrass.StreamManager` come dipendenza del componente quando importi la funzione. Quando distribuisci il componente stream manager, specifica i parametri dello stream manager da impostare per i dispositivi principali di destinazione. Il ruolo di scambio di token del dispositivo principale deve consentire al dispositivo principale di accedere alle Cloud AWS destinazioni utilizzate con stream manager. Per ulteriori informazioni, consulta Stream manager.

- Segreti locali

  Se la tua funzione Lambda utilizza segreti locali, specifica `aws.greengrass.SecretManager` come dipendenza del componente quando importi la funzione. Quando distribuisci il componente Secret Manager, specifica le risorse segrete da distribuire sui dispositivi principali di destinazione. Il ruolo di scambio di token del dispositivo principale deve consentire al dispositivo principale di recuperare le risorse segrete da distribuire. Per ulteriori informazioni, consulta Gestore segreto.

Quando distribuisci il componente della funzione Lambda, configuralo in modo che disponga di una politica di autorizzazione IPC che conceda il permesso di utilizzare l'operazione IPC GetSecretValue nella V2. SDK per dispositivi AWS IoT

- Ombre locali

  Se la funzione Lambda interagisce con le ombre locali, è necessario aggiornare il codice della funzione Lambda per utilizzare la V2. SDK per dispositivi AWS IoT È inoltre necessario specificare `aws.greengrass.ShadowManager` come componente la dipendenza quando si importa la funzione. Per ulteriori informazioni, consulta Interagisci con le ombre dei dispositivi.

  Quando distribuisci il componente della funzione Lambda, configuralo in modo che disponga di una politica di autorizzazione IPC che conceda l'autorizzazione all'uso delle operazioni IPC shadow nella V2. SDK per dispositivi AWS IoT

- Sottoscrizioni
  - Se la tua funzione Lambda sottoscrive i messaggi da una fonte cloud, specifica tali sottoscrizioni come fonti di eventi quando importi la funzione.
  - Se la funzione Lambda sottoscrive i messaggi di un'altra funzione Lambda o se la funzione Lambda pubblica messaggi verso o AWS IoT Core altre funzioni Lambda, configura e distribuisci il componente legacy del router di abbonamento quando distribuisci la funzione Lambda. Quando distribuisci il componente legacy del router di sottoscrizione, specifica gli abbonamenti utilizzati dalla funzione Lambda.

    > **Note**
    >
    > Il componente legacy del router di abbonamento è richiesto solo se la funzione Lambda utilizza la `publish()` funzione nel AWS IoT Greengrass Core SDK. Se aggiorni il codice della funzione Lambda per utilizzare l'interfaccia di comunicazione tra processi (IPC) nella SDK per dispositivi AWS IoT V2, non è necessario implementare il componente legacy del router di abbonamento. Per ulteriori informazioni, consulta i seguenti servizi di comunicazione tra processi:
    >
    > - Pubblicare/sottoscrivere messaggi locali
    > - AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

  - Se la tua funzione Lambda sottoscrive i messaggi provenienti da dispositivi connessi localmente, specifica tali sottoscrizioni come fonti di eventi quando importi la funzione. È inoltre necessario

configurare e distribuire il componente bridge MQTT per inoltrare i messaggi dai dispositivi collegati agli argomenti di pubblicazione/sottoscrizione locali specificati come fonti di eventi.

- Se la funzione Lambda pubblica messaggi su dispositivi connessi locali, è necessario aggiornare il codice della funzione Lambda per utilizzare la SDK per dispositivi AWS IoT V2 per pubblicare messaggi di pubblicazione/sottoscrizione locali. È inoltre necessario configurare e distribuire il componente bridge MQTT per inoltrare i messaggi dal broker di messaggi di pubblicazione/sottoscrizione locale ai dispositivi collegati.

- Volumi e dispositivi locali

  Se la tua funzione Lambda containerizzata accede a volumi o dispositivi locali, specifica tali volumi e dispositivi quando importi la funzione Lambda. Questa funzionalità non richiede una dipendenza dai componenti.

Per ulteriori informazioni, consulta Esegui AWS Lambda funzioni.

Usa connettori V1

È possibile distribuire componenti AWS forniti che offrono le stesse funzionalità di alcuni connettori. AWS IoT Greengrass Quando si crea la distribuzione, è possibile configurare i parametri dei connettori.

I seguenti AWS IoT Greengrass V2 componenti forniscono la funzionalità del connettore Greengrass V1:

- CloudWatch componente metrico
- AWS IoT Device Defender componente
- Componente Firehose
- Componente dell'adattatore di protocollo Modbus-RTU
- Componente Amazon SNS

Esegui contenitori Docker

AWS IoT Greengrass V2 non fornisce un componente per sostituire direttamente il connettore di distribuzione delle applicazioni Docker V1. Tuttavia, puoi utilizzare il componente Docker Application Manager per scaricare immagini Docker e quindi creare componenti personalizzati che eseguono contenitori Docker a partire dalle immagini scaricate. Per ulteriori informazioni, consultare Esegui un contenitore Docker e Gestore di applicazioni Docker.

Esegui l'inferenza dell'apprendimento automatico

AWS IoT Greengrass V2 fornisce un componente Amazon SageMaker Edge Manager che installa l'agente Amazon SageMaker Edge Manager e consente di utilizzare modelli SageMaker NEO compilati come componenti del modello sui dispositivi core Greengrass. AWS IoT Greengrass V2 fornisce anche componenti che installano Deep Learning Runtime e TensorFlow Lite sul tuo dispositivo. È possibile utilizzare i componenti di inferenza e modelli DLR e TensorFlow Lite corrispondenti per eseguire la classificazione delle immagini dei campioni e l'inferenza per il rilevamento degli oggetti. Per utilizzare altri framework di machine learning, come MXNet e TensorFlow, è possibile sviluppare componenti personalizzati che utilizzano questi framework.

Dispositivi Connect V1 Greengrass

I dispositivi collegati AWS IoT Greengrass V1 sono denominati dispositivi client in. AWS IoT Greengrass V2 AWS IoT Greengrass V2 il supporto per i dispositivi client è retrocompatibile con AWS IoT Greengrass V1, quindi è possibile connettere i dispositivi client V1 ai dispositivi core V2 senza modificare il codice dell'applicazione. Per consentire ai dispositivi client di connettersi a un dispositivo core V2, distribuisci i componenti Greengrass che abilitano il supporto dei dispositivi client e associa i dispositivi client al dispositivo principale. Per inoltrare messaggi tra i dispositivi client, il servizio AWS IoT Core cloud e i componenti Greengrass (incluse le funzioni Lambda), implementate e configurate il componente bridge MQTT. È possibile implementare il componente IP Detector per rilevare automaticamente le informazioni di connettività oppure gestire manualmente gli endpoint. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

Abilita il servizio shadow locale

In AWS IoT Greengrass V2, il servizio shadow locale è implementato dal componente shadow manager AWS fornito. AWS IoT Greengrass V2 include anche il supporto per le ombre denominate. Per consentire ai componenti di interagire con le ombre locali e sincronizzare gli stati shadow AWS IoT Core, configura e distribuisci il componente Shadow Manager e utilizza le operazioni IPC shadow nel codice del componente. Per ulteriori informazioni, consulta Interagisci con le ombre dei dispositivi.

Integrazione con AWS IoT SiteWise

Se utilizzi il tuo dispositivo core V1 come AWS IoT SiteWise gateway, segui le istruzioni per configurare il tuo nuovo dispositivo core V2 come AWS IoT SiteWise gateway. AWS IoT SiteWise fornisce uno script di installazione che distribuisce i AWS IoT SiteWise componenti per voi.

Fase 3: testate le vostre applicazioni AWS IoT Greengrass V2

Dopo aver creato e distribuito i componenti V2 sul nuovo dispositivo core V2, verifica che le applicazioni soddisfino le tue aspettative. Puoi controllare i registri del dispositivo per visualizzare i messaggi di output standard (stdout) e di errore standard (stderr) dei componenti. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Se hai distribuito la CLI Greengrass sul dispositivo principale, puoi utilizzarla per eseguire il debug dei componenti e delle relative configurazioni. Per ulteriori informazioni, consulta Comandi della CLI di Greengrass.

Dopo aver verificato che le applicazioni funzionino su un dispositivo core V2, puoi distribuire i componenti Greengrass dell'applicazione su altri dispositivi principali. Se hai sviluppato componenti personalizzati che eseguono processi nativi o contenitori Docker, devi prima pubblicare tali componenti sul AWS IoT Greengrass servizio per distribuirli su altri dispositivi principali.

Aggiorna i dispositivi core Greengrass V1 a Greengrass V2

Dopo aver verificato che le applicazioni e i componenti funzionino su un dispositivo AWS IoT Greengrass V2 principale, puoi installare il software AWS IoT Greengrass Core v2.x sui dispositivi che attualmente eseguono la versione 1.x, come i dispositivi di produzione. Quindi, distribuisci i componenti Greengrass V2 per eseguire le tue applicazioni Greengrass sui dispositivi.

Per aggiornare una flotta di dispositivi dalla V1 alla V2, completa questi passaggi per ogni dispositivo da aggiornare. Puoi utilizzare i gruppi di oggetti per distribuire i componenti V2 su una flotta di dispositivi principali.

> **Tip**
>
> Ti consigliamo di creare uno script per automatizzare il processo di aggiornamento per una flotta di dispositivi. Se utilizzi AWS Systems Managerper gestire la tua flotta, puoi utilizzare Systems Manager per eseguire lo script su ogni dispositivo per aggiornare la tua flotta dalla V1 alla V2.
> Puoi contattare il tuo rappresentante AWS Enterprise Support per domande su come automatizzare al meglio il processo di aggiornamento.

Fase 1: Installare il software AWS IoT Greengrass Core v2.x

Scegli tra le seguenti opzioni per installare il software AWS IoT Greengrass Core v2.x su un dispositivo core V1:

- Effettua l'upgrade in meno passaggi

  Per eseguire l'aggiornamento in meno passaggi, puoi disinstallare il software v1.x prima di installare il software v2.x.

- Esegui l'upgrade con tempi di inattività minimi

  Per eseguire l'aggiornamento con tempi di inattività minimi, puoi installare entrambe le versioni del software AWS IoT Greengrass Core contemporaneamente. Dopo aver installato il software AWS IoT Greengrass Core v2.x e verificato che le applicazioni V2 funzionino correttamente, disinstallate il AWS IoT Greengrass software Core v1.x. Prima di scegliere questa opzione, considerate la RAM aggiuntiva necessaria per eseguire entrambe le versioni del software AWS IoT Greengrass Core contemporaneamente.

Disinstalla AWS IoT Greengrass Core v1.x prima di installare la v2.x

Se desideri eseguire l'aggiornamento in sequenza, disinstalla il software AWS IoT Greengrass Core v1.x prima di installare la v2.x sul tuo dispositivo.

Per disinstallare il software Core v1.x AWS IoT Greengrass

1. Se il software AWS IoT Greengrass Core v1.x è in esecuzione come servizio, è necessario arrestare, disabilitare e rimuovere il servizio.

   a. Arrestare il servizio AWS IoT Greengrass Core software v1.x in esecuzione.

   ```
   sudo systemctl stop greengrass
   ```

   b. Attendi che il servizio si interrompa. È possibile utilizzare il `list` comando per verificare lo stato del servizio.

   ```
   sudo systemctl list-units --type=service | grep greengrass
   ```

   c. Disabilita il servizio.

```
sudo systemctl disable greengrass
```

d. Rimuovi il servizio.

```
sudo rm /etc/systemd/system/greengrass.service
```

2. Se il software AWS IoT Greengrass Core v1.x non è in esecuzione come servizio, utilizzate il seguente comando per arrestare il demone. Sostituisci *greengrass-root* con il nome della tua cartella principale Greengrass. Il percorso predefinito è `/greengrass`.

```
cd /greengrass-root/ggc/core/
sudo ./greengrassd stop
```

3. (Facoltativo) Eseguite il backup della cartella principale di Greengrass e, se applicabile, della cartella di scrittura personalizzata, in un'altra cartella sul dispositivo.

   a. Usa il seguente comando per copiare la cartella principale Greengrass corrente in una cartella diversa, quindi rimuovere la cartella principale.

```
sudo cp -r /greengrass-root /path/to/greengrass-backup
rm -rf /greengrass-root
```

   b. Utilizzate il seguente comando per spostare la cartella di scrittura in una cartella diversa, quindi rimuovete la cartella di scrittura.

```
sudo cp -r /write-directory /path/to/write-directory-backup
rm -rf /write-directory
```

È quindi possibile utilizzare le istruzioni di installazione AWS IoT Greengrass V2 per installare il software sul dispositivo.

> **Tip**
>
> Per riutilizzare l'identità di un dispositivo principale durante la migrazione dalla V1 alla V2, segui le istruzioni per installare il software AWS IoT Greengrass Core con provisioning manuale. Rimuovi innanzitutto il software di base V1 dal dispositivo, quindi riutilizza l'AWS

IoToggetto e il certificato del dispositivo principale V1 e aggiorna le AWS IoT politiche del certificato per concedere le autorizzazioni richieste dal software v2.x.

Installa il software AWS IoT Greengrass Core v2.x su un dispositivo che esegue già la versione 1.x

Se installi il software AWS IoT Greengrass Core v2.x su un dispositivo su cui è già in esecuzione il software AWS IoT Greengrass Core v1.x, tieni presente quanto segue:

- Il nome dell'AWS IoToggetto per il dispositivo core V2 deve essere univoco. Non utilizzare lo stesso nome del dispositivo principale V1.
- Le porte utilizzate per il software AWS IoT Greengrass Core v2.x devono essere diverse dalle porte utilizzate per la v1.x.
  - Configura lo stream manager V1 per utilizzare una porta diversa dalla 8088. Per ulteriori informazioni, consulta Configurare lo stream manager.
  - Configurate il broker MQTT V1 per utilizzare una porta diversa dalla 8883. Per ulteriori informazioni, vedere Configurazione della porta MQTT per la messaggistica locale.
- AWS IoT Greengrass V2non offre la possibilità di rinominare il servizio di sistema Greengrass. Se si esegue Greengrass come servizio di sistema, è necessario effettuare una delle seguenti operazioni per evitare conflitti tra i nomi dei servizi di sistema:
  - Rinomina il servizio Greengrass per la v1.x prima di installare la v2.x.
  - Installa il software AWS IoT Greengrass Core v2.x senza un servizio di sistema, quindi configura manualmente il software come servizio di sistema con un nome diverso da. `greengrass`

  Per rinominare il servizio Greengrass per la versione 1.x

  1. Interrompere il servizio AWS IoT Greengrass Core software v1.x.

     ```
     sudo systemctl stop greengrass
     ```

  2. Attendi che il servizio si interrompa. L'interruzione del servizio può richiedere fino a qualche minuto. È possibile utilizzare il `list-units` comando per verificare se il servizio è stato interrotto.

     ```
     sudo systemctl list-units --type=service | grep greengrass
     ```

3. Disabilita il servizio.

```
sudo systemctl disable greengrass
```

4. Rinomina il servizio.

```
sudo mv /etc/systemd/system/greengrass.service /etc/systemd/system/greengrass-v1.service
```

5. Ricarica il servizio e avvialo.

```
sudo systemctl daemon-reload
sudo systemctl reset-failed
sudo systemctl enable greengrass-v1
sudo systemctl start greengrass-v1
```

È quindi possibile utilizzare le istruzioni di installazione AWS IoT Greengrass V2 per installare il software sul dispositivo.

> **Tip**
>
> Per riutilizzare l'identità di un dispositivo principale durante la migrazione dalla V1 alla V2, segui le istruzioni per installare il software AWS IoT Greengrass Core con provisioning manuale. Rimuovi innanzitutto il software di base V1 dal dispositivo, quindi riutilizza l'AWS IoToggetto e il certificato del dispositivo principale V1 e aggiorna le AWS IoT politiche del certificato per concedere le autorizzazioni richieste dal software v2.x.

Fase 2: Implementazione dei componenti sui dispositivi principali AWS IoT Greengrass V2

Dopo aver installato il software AWS IoT Greengrass Core v2.x sul dispositivo, crea una distribuzione che includa le seguenti risorse. Per distribuire componenti su una flotta di dispositivi simili, crea una distribuzione per un gruppo di oggetti che contiene tali dispositivi.

- Componenti della funzione Lambda creati a partire dalle funzioni Lambda V1. Per ulteriori informazioni, consulta Esegui AWS Lambda funzioni.
- Se utilizzi abbonamenti V1, il componente legacy del router di sottoscrizione.

- Se usi stream manager, il componente stream manager. Per ulteriori informazioni, consulta Gestisci i flussi di dati sui dispositivi core Greengrass.
- Se usi i segreti locali, il componente del gestore segreto.
- Se si utilizzano connettori V1, i componenti del connettore AWS forniti.
- Se utilizzi contenitori Docker, il componente Docker Application Manager. Per ulteriori informazioni, consulta Esegui un contenitore Docker.
- Se utilizzi l'inferenza dell'apprendimento automatico, i componenti per il supporto dell'apprendimento automatico. Per ulteriori informazioni, consulta Esecuzione dell'inferenza di Machine Learning.
- Se si utilizzano dispositivi connessi, i componenti per i dispositivi client supportano. È inoltre necessario abilitare il supporto per i dispositivi client e associare i dispositivi client al dispositivo principale. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.
- Se utilizzate Device Shadows, il componente Shadow Manager. Per ulteriori informazioni, consulta Interagisci con le ombre dei dispositivi.
- Se carichi i log dai dispositivi core di Greengrass su CloudWatch Amazon Logs, il componente di gestione dei log. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.
- Se esegui l'integrazione conAWS IoT SiteWise, segui le istruzioni per configurare il dispositivo principale V2 come gateway. AWS IoT SiteWise AWS IoT SiteWisefornisce uno script di installazione che distribuisce i AWS IoT SiteWise componenti per voi.
- componenti definiti dall'utente che avete sviluppato per implementare funzionalità personalizzate.

Per informazioni sulla creazione e la revisione delle distribuzioni, vedere. Implementazione AWS IoT Greengrass dei componenti sui dispositivi

Tutorial: Nozioni di base su AWS IoT Greengrass V2

Puoi completare questo tutorial introduttivo per apprendere le funzionalità di base diAWS IoT Greengrass V2. In questo tutorial, esegui quanto indicato di seguito:

1. Installa e configura il software AWS IoT Greengrass Core su un dispositivo Linux, come un Raspberry Pi o un dispositivo Windows. Questo dispositivo è un dispositivo principale Greengrass.
2. Sviluppa un componente Hello World sul tuo dispositivo principale Greengrass. I componenti sono moduli software che funzionano sui dispositivi core Greengrass.
3. Carica quel componente AWS IoT Greengrass V2 in. Cloud AWS
4. Distribuisci quel componente dal tuo Cloud AWS dispositivo principale Greengrass.

> **Note**
>
> Questo tutorial descrive come configurare un ambiente di sviluppo ed esplorare le funzionalità di. AWS IoT Greengrass Per ulteriori informazioni su come impostare e configurare i dispositivi di produzione, consulta quanto segue:
>
> - Configurazione dei dispositivi AWS IoT Greengrass principali
> - Installare il software AWS IoT Greengrass Core.

Puoi aspettarti di dedicare dai 20 ai 30 minuti per questo tutorial.

Argomenti

Prerequisiti

Per completare questo tutorial delle nozioni di base occorrono i seguenti requisiti:

- Un Account AWS. Se non lo hai, consultare Fase 1: Impostazione di un account AWS.
- L'uso di un Regione AWSventilatore che supportiAWS IoT Greengrass V2. Per l'elenco delle regioni supportate, consulta Endpoint e quote AWS IoT Greengrass V2 in Riferimenti generali di AWS.
- Un utente AWS Identity and Access Management (IAM) con autorizzazioni di amministratore.
- Un dispositivo da configurare come dispositivo principale Greengrass, ad esempio un Raspberry Pi con sistema operativo Raspberry Pi (precedentemente chiamato Raspbian) o un dispositivo Windows 10. È necessario disporre delle autorizzazioni di amministratore su questo dispositivo o della possibilità di acquisire privilegi di amministratore, ad esempio tramite. `sudo` Questo dispositivo deve disporre di una connessione Internet.

  Puoi anche scegliere di utilizzare un dispositivo diverso che soddisfi i requisiti per installare ed eseguire il software AWS IoT Greengrass Core. Per ulteriori informazioni, consulta Piattaforme supportate e requisiti.

  Se il tuo computer di sviluppo soddisfa questi requisiti, puoi configurarlo come dispositivo principale Greengrass in questo tutorial.
- Python 3.5 o successivo installato per tutti gli utenti sul dispositivo e aggiunto alla variabile di `PATH` ambiente. Su Windows, è inoltre necessario che Python Launcher per Windows sia installato per tutti gli utenti.

  > **Important**
  >
  > In Windows, Python non viene installato per tutti gli utenti per impostazione predefinita. Quando installi Python, devi personalizzare l'installazione per configurarla affinché il software AWS IoT Greengrass Core esegua gli script Python. Ad esempio, se usi il programma di installazione grafico di Python, procedi come segue:
  >
  > 1. Seleziona Installa il programma di avvio per tutti gli utenti (consigliato).
  > 2. Scegli Customize installation.
  > 3. Scegli Next.
  > 4. Seleziona Install for all users.
  > 5. Seleziona Add Python to environment variables.

6. Scegli Installa.

Per ulteriori informazioni, consulta Usare Python su Windows nella documentazione di Python 3.

- AWS Command Line Interface(AWS CLI) installato e configurato con credenziali sul computer di sviluppo e sul dispositivo. Assicurati di utilizzarle Regione AWS per configurarle AWS CLI sul tuo computer di sviluppo e sul tuo dispositivo. Per utilizzarlo AWS IoT Greengrass V2 conAWS CLI, è necessario disporre di una delle seguenti versioni o successive:
  - Versione minima AWS CLI V1: v1.18.197
  - Versione minima AWS CLI V2: v2.1.11

> **Tip**
>
> Puoi eseguire il seguente comando per verificare la versione di cui disponiAWS CLI.
>
> ```
> aws --version
> ```

Per ulteriori informazioni, vedere Installazione, aggiornamento e disinstallazione AWS CLI e configurazione di AWS CLI nella Guida per l'AWS Command Line Interfaceutente.

> **Note**
>
> Se utilizzi un dispositivo ARM a 32 bit, come un Raspberry Pi con un sistema operativo a 32 bit, installa V1. AWS CLI AWS CLI La versione 2 non è disponibile per i dispositivi ARM a 32 bit. Per ulteriori informazioni, vedere Installazione, aggiornamento e disinstallazione della AWS CLI versione 1.

Fase 1: Impostazione di un account AWS

Registrarsi per creare un Account AWS

Se non disponi di un Account AWS, completa la procedura seguente per crearne uno.

Per registrarsi a un Account AWS

1. Apri la pagina all'indirizzo https://portal.aws.amazon.com/billing/signup.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

   Durante la registrazione di un Account AWS, viene creato un Utente root dell'account AWS. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWSnell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente amministrativo e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

Al termine del processo di registrazione, riceverai un'e-mail di conferma da AWS. È possibile visualizzare l'attività corrente dell'account e gestire l'account in qualsiasi momento accedendo all'indirizzo https://aws.amazon.com/ e selezionando Il mio account.

Creazione di un utente amministratore

Dopo aver effettuato la registrazione di un Account AWS, proteggi Utente root dell'account AWS, abilita AWS IAM Identity Center e crea un utente amministratore in modo da non utilizzare l'utente root per le attività quotidiane.

Protezione dell'Utente root dell'account AWS

1. Accedi alla AWS Management Console come proprietario dell'account scegliendo Utente root e immettendo l'indirizzo email del Account AWS. Nella pagina successiva, inserisci la password.

   Per informazioni sull'accesso utilizzando un utente root, consulta la pagina Accesso come utente root della Guida per l'utente di Accedi ad AWS.
2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

   Per ricevere istruzioni, consulta Abilitazione di un dispositivo MFA virtuale per l'utente root dell'Account AWS (console) nella Guida per l'utente IAM.

Creazione di un utente amministratore

1. Abilita IAM Identity Center

Per istruzioni, consulta Abilitazione di AWS IAM Identity Center nella Guida per l'utente di AWS IAM Identity Center.

2. In Centro identità AWS IAM, assegna l'accesso amministrativo a un utente amministrativo.

   Per un tutorial sull'utilizzo di IAM Identity Center directory come origine di identità, consulta Configure user access with the default IAM Identity Center directory nella Guida per l'utente di AWS IAM Identity Center.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

  Per informazioni sull'accesso utilizzando un utente IAM Identity Center, consulta Accedere al portale di accesso AWS nella Guida per l'utente Accedi ad AWS.

Fase 2: Configurare l'ambiente

Segui i passaggi di questa sezione per configurare un dispositivo Linux o Windows da utilizzare come dispositivo AWS IoT Greengrass principale.

Configura un dispositivo Linux (Raspberry Pi)

Questi passaggi presuppongono l'utilizzo di un Raspberry Pi con sistema operativo Raspberry Pi. Se utilizzi un dispositivo o un sistema operativo diverso, consulta la documentazione pertinente del tuo dispositivo.

Per configurare un Raspberry Pi per AWS IoT Greengrass V2

1. Abilita SSH sul tuo Raspberry Pi per connetterti in remoto ad esso. Per ulteriori informazioni, consulta SSH (Secure shell) nella documentazione di Raspberry Pi.
2. Trova l'indirizzo IP del tuo Raspberry Pi per connetterti ad esso con SSH. Per farlo, puoi eseguire il seguente comando sul tuo Raspberry Pi.

   ```
   hostname -I
   ```

3. Connect al tuo Raspberry Pi con SSH.

Sul tuo computer di sviluppo, esegui il seguente comando. Sostituisci il nome *utente* con il nome dell'utente a cui accedere e sostituiscilo *pi-ip-address*con l'indirizzo IP che hai trovato nel passaggio precedente.

```
ssh username@pi-ip-address
```

> **Important**
>
> Se il tuo computer di sviluppo utilizza una versione precedente di Windows, potresti non avere il `ssh` comando oppure potresti averlo `ssh` ma non riesci a connetterti al tuo Raspberry Pi. Per connetterti al tuo Raspberry Pi, puoi installare e configurare PuTTY, un client SSH open source gratuito. Consulta la documentazione di PuTTY per connetterti al tuo Raspberry Pi.

4. Installa il runtime Java, necessario per l'esecuzione AWS IoT Greengrass del software Core. Sul tuo Raspberry Pi, usa i seguenti comandi per installare Java 11.

```
sudo apt install default-jdk
```

Al termine dell'installazione, esegui il comando seguente per verificare che Java sia in esecuzione sul tuo Raspberry Pi.

```
java -version
```

Il comando stampa la versione di Java in esecuzione sul dispositivo. L'output potrebbe essere simile all'esempio seguente.

```
openjdk version "11.0.9.1" 2020-11-04
OpenJDK Runtime Environment (build 11.0.9.1+1-post-Debian-1deb10u2)
OpenJDK 64-Bit Server VM (build 11.0.9.1+1-post-Debian-1deb10u2, mixed mode)
```

> **Suggerimento: imposta i parametri del kernel su un Raspberry Pi**
>
> Se il tuo dispositivo è un Raspberry Pi, puoi completare i seguenti passaggi per visualizzare e aggiornare i parametri del kernel Linux:

1. Apri il file `/boot/cmdline.txt`. Questo file specifica i parametri del kernel Linux da applicare all'avvio del Raspberry Pi.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per aprire il file.

   ```
   sudo nano /boot/cmdline.txt
   ```

2. Verificate che il `/boot/cmdline.txt` file contenga i seguenti parametri del kernel. Il `systemd.unified_cgroup_hierarchy=0` parametro specifica di utilizzare cgroups v1 invece di cgroups v2.

   ```
   cgroup_enable=memory cgroup_memory=1 systemd.unified_cgroup_hierarchy=0
   ```

   Se il `/boot/cmdline.txt` file non contiene questi parametri o contiene questi parametri con valori diversi, aggiorna il file in modo che contenga questi parametri e valori.

3. Se hai aggiornato il `/boot/cmdline.txt` file, riavvia Raspberry Pi per applicare le modifiche.

   ```
   sudo reboot
   ```

Configura un dispositivo Linux (altro)

Per configurare un dispositivo Linux per AWS IoT Greengrass V2

1. Installa il runtime Java, necessario per l'esecuzione del software AWS IoT Greengrass Core. Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva. I seguenti comandi mostrano come installare OpenJDK sul tuo dispositivo.

   - Per le distribuzioni basate su Debian o basate su Ubuntuu:

     ```
     sudo apt install default-jdk
     ```

   - Per le distribuzioni basate su Red Hat:

```
sudo yum install java-11-openjdk-devel
```

- Per Amazon Linux 2:

```
sudo amazon-linux-extras install java-openjdk11
```

- Per Amazon Linux 2023:

```
sudo dnf install java-11-amazon-corretto -y
```

Al termine dell'installazione, esegui il comando seguente per verificare che Java sia in esecuzione sul tuo dispositivo Linux.

```
java -version
```

Il comando stampa la versione di Java in esecuzione sul dispositivo. Ad esempio, su una distribuzione basata su Debian, l'output potrebbe essere simile all'esempio seguente.

```
openjdk version "11.0.9.1" 2020-11-04
OpenJDK Runtime Environment (build 11.0.9.1+1-post-Debian-1deb10u2)
OpenJDK 64-Bit Server VM (build 11.0.9.1+1-post-Debian-1deb10u2, mixed mode)
```

2. (Facoltativo) Crea l'utente e il gruppo di sistema predefiniti che eseguono i componenti sul dispositivo. Puoi anche scegliere di lasciare che il programma di installazione del software AWS IoT Greengrass Core crei questo utente e gruppo durante l'installazione con l'argomento `--component-default-user` installer. Per ulteriori informazioni, consulta Argomenti dell'installatore.

```
sudo useradd --system --create-home ggc_user
sudo groupadd --system ggc_group
```

3. Verificate che l'utente che esegue il software AWS IoT Greengrass Core (in genere`root`) sia autorizzato a funzionare `sudo` con qualsiasi utente e gruppo.

   a. Eseguite il comando seguente per aprire il `/etc/sudoers` file.

```
sudo visudo
```

b. Verificate che l'autorizzazione per l'utente sia simile all'esempio seguente.

```
root    ALL=(ALL:ALL) ALL
```

4. (Facoltativo) Per eseguire funzioni Lambda containerizzate, è necessario abilitare cgroups v1 e abilitare e montare i cgroup di memoria e dispositivi. Se non intendi eseguire funzioni Lambda containerizzate, puoi saltare questo passaggio.

   Per abilitare queste opzioni di cgroups, avvia il dispositivo con i seguenti parametri del kernel Linux.

   ```
   cgroup_enable=memory cgroup_memory=1 systemd.unified_cgroup_hierarchy=0
   ```

   Per informazioni sulla visualizzazione e l'impostazione dei parametri del kernel per il tuo dispositivo, consulta la documentazione del tuo sistema operativo e del boot loader. Segui le istruzioni per impostare in modo permanente i parametri del kernel.

5. Installa tutte le altre dipendenze richieste sul tuo dispositivo come indicato dall'elenco dei requisiti in. Requisiti per il dispositivo

Configura un dispositivo Windows

Per configurare un dispositivo Windows per AWS IoT Greengrass V2

1. Installa il runtime Java, necessario per l'esecuzione del software AWS IoT Greengrass Core. Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva.
2. Controlla se Java è disponibile nella variabile di sistema PATH e aggiungilo in caso contrario. L' LocalSystem account esegue il software AWS IoT Greengrass Core, quindi è necessario aggiungere Java alla variabile di sistema PATH anziché alla variabile utente PATH per l'utente. Esegui questa operazione:

   a. Premi il tasto Windows per aprire il menu di avvio.

   b. Digita **`environment variables`** per cercare le opzioni di sistema dal menu di avvio.

   c. Nei risultati della ricerca del menu di avvio, scegli Modifica le variabili di ambiente di sistema per aprire la finestra delle proprietà del sistema.

   d. Scegli le variabili di ambiente... per aprire la finestra Variabili d'ambiente.

e. In Variabili di sistema, seleziona Percorso, quindi scegli Modifica. Nella finestra Modifica variabile di ambiente, puoi visualizzare ogni percorso su una riga separata.

f. Controlla se è presente il percorso della `bin` cartella di installazione di Java. Il percorso potrebbe essere simile all'esempio seguente.

```
C:\\Program Files\\Amazon Corretto\\jdk11.0.13_8\\bin
```

g. Se la `bin` cartella di installazione Java non è presente in Path, scegliete Nuovo per aggiungerla, quindi scegliete OK.

3. Aprite il prompt dei comandi di Windows (`cmd.exe`) come amministratore.

4. Crea l'utente predefinito nell' LocalSystem account sul dispositivo Windows. Sostituisci *`la password`* con una password sicura.

```
net user /add ggc_user password
```

> **Tip**
>
> A seconda della configurazione di Windows, la password dell'utente potrebbe essere impostata per scadere in date future. Per garantire che le tue applicazioni Greengrass continuino a funzionare, tieni traccia della scadenza della password e aggiornala prima che scada. Puoi anche impostare la password dell'utente in modo che non scada mai.
>
> - Per verificare la scadenza di un utente e della relativa password, esegui il comando seguente.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Per impostare la password di un utente in modo che non scada mai, esegui il comando seguente.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```
>
> - Se utilizzi Windows 10 o versioni successive in cui il wmiccomando è obsoleto, esegui il comando seguente. PowerShell
>
>   ```
>   Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
>    'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
>   ```

5. Scarica e installa l'PsExecutilità di Microsoft sul dispositivo.
6. Utilizzate l' PsExec utilità per memorizzare il nome utente e la password per l'utente predefinito nell'istanza di Credential Manager per l' LocalSystem account. Sostituisci la *password* con la password dell'utente impostata in precedenza.

   ```
   psexec -s cmd /c cmdkey /generic:ggc_user /user:ggc_user /pass:password
   ```

   Se si PsExec License Agreementapre, scegli Acceptdi accettare la licenza ed esegui il comando.

   > **Note**
   >
   > Sui dispositivi Windows, l' LocalSystem account esegue il Greengrass nucleus ed è necessario utilizzare l' PsExec utilità per memorizzare le informazioni utente predefinite nell'account. LocalSystem L'utilizzo dell'applicazione Credential Manager archivia queste informazioni nell'account Windows dell'utente attualmente connesso, anziché nell'account. LocalSystem

Fase 3: installare il softwareAWS IoT Greengrass Core

Segui i passaggi in questa sezione per configurare il tuo Raspberry Pi come dispositivoAWS IoT Greengrass principale da utilizzare per lo sviluppo locale. In questa sezione, scarichi ed esegui un programma di installazione che esegue le seguenti operazioni per configurare il softwareAWS IoT Greengrass Core per il tuo dispositivo:

- Installa il componente Greengrass nucleus. Il nucleo è un componente obbligatorio ed è il requisito minimo per eseguire il softwareAWS IoT Greengrass Core su un dispositivo. Per ulteriori informazioni, consulta la pagina relativa ai Componenti del nucleo Greengrass.
- Registra il dispositivo comeAWS IoT oggetto e scarica un certificato digitale che consente al dispositivo di connettersiAWS. Per ulteriori informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass.
- Aggiunge l'AWS IoTelemento del dispositivo a un gruppo di oggetti, che è un gruppo o un parco diAWS IoT cose. I gruppi di cose consentono di gestire flotte di dispositivi principali Greengrass. Quando distribuisci componenti software sui tuoi dispositivi, puoi scegliere di distribuirli su singoli dispositivi o su gruppi di dispositivi. Per ulteriori informazioni, consulta Gestire i dispositivi conAWS IoT nella Guida per gliAWS IoT Core sviluppatori.

- Crea il ruolo IAM che consente al dispositivo principale Greengrass di interagire conAWS i servizi. Per impostazione predefinita, questo ruolo consente al dispositivo di interagireAWS IoT e inviare registri ad AmazonCloudWatch Logs. Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.
- Installa l'interfaccia a riga diAWS IoT Greengrass comando (`greengrass-cli`), che puoi usare per testare i componenti personalizzati sviluppati sul dispositivo principale. Per ulteriori informazioni, consulta Interfaccia a riga di comando Greengrass.

Installare il software AWS IoT Greengrass Core (console)

1. Accedi alla console AWS IoT Greengrass.
2. In Inizia a usare Greengrass, scegli Configura un dispositivo principale.
3. Nel Passaggio 1: registra un dispositivo principale Greengrass, per il nome del dispositivo Core, inserisci il nome dell'AWS IoToggetto per il tuo dispositivo principale Greengrass. Se l'oggetto non esiste, il programma di installazione lo crea.
4. Nel Passaggio 2: Aggiungi a un gruppo di oggetti per applicare una distribuzione continua, per Thing group, scegli il AWS IoT gruppo di oggetti a cui desideri aggiungere il dispositivo principale.

   - Se si seleziona Inserisci un nuovo nome di gruppo, in Thing group name, inserisci il nome del nuovo gruppo da creare. Il programma di installazione crea automaticamente il nuovo gruppo.
   - Se si seleziona Seleziona un gruppo esistente, in Nome gruppo di oggetti, scegli il gruppo esistente che desideri utilizzare.
   - Se si seleziona Nessun gruppo, il programma di installazione non aggiunge il dispositivo principale a un gruppo di oggetti.
5. Nel Passaggio 3: Installare il software Greengrass Core, completare i seguenti passaggi.

   a. Scegli il sistema operativo del tuo dispositivo principale: Linux o Windows.

   b. Fornisci AWS le tue credenziali al dispositivo in modo che l'installatore possa fornire le risorse AWS IoT e IAM per il tuo dispositivo principale. Per aumentare la sicurezza, ti consigliamo di ottenere credenziali temporanee per un ruolo IAM che consenta solo le autorizzazioni minime necessarie per il provisioning. Per ulteriori informazioni, consulta Policy IAM minima per l'installatore per il provisioning delle risorse.

> **Note**
>
> Il programma di installazione non salva né archivia le tue credenziali.

Sul dispositivo, esegui una delle seguenti operazioni per recuperare le credenziali e renderle disponibili al programma di installazione del AWS IoT Greengrass software Core:

- (Consigliato) Utilizza credenziali temporanee da AWS IAM Identity Center

  i. Fornisci l'ID della chiave di accesso, la chiave di accesso segreta e il token di sessione dall'IAM Identity Center. Per ulteriori informazioni, consulta Aggiornamento manuale delle credenziali in Acquisizione e aggiornamento delle credenziali temporanee nella guida per l'utente di IAM Identity Center.

  ii. Esegui i seguenti comandi per fornire le credenziali al software Core. AWS IoT Greengrass

  Linux or Unix

  ```
  export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
  export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/
  bPxRfiCYEXAMPLEKEY
  export AWS_SESSION_TOKEN=AQoDYXdzEJr1K...o5OytwEXAMPLE=
  ```

  Windows Command Prompt (CMD)

  ```
  set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
  set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
  set AWS_SESSION_TOKEN=AQoDYXdzEJr1K...o5OytwEXAMPLE=
  ```

  PowerShell

  ```
  $env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
  $env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/
  bPxRfiCYEXAMPLEKEY"
  $env:AWS_SESSION_TOKEN="AQoDYXdzEJr1K...o5OytwEXAMPLE="
  ```

- Utilizza credenziali di sicurezza temporanee da un ruolo IAM:

i. Fornisci l'ID della chiave di accesso, la chiave di accesso segreta e il token di sessione da un ruolo IAM che assumi. Per ulteriori informazioni su come recuperare queste credenziali, consulta la sezione Richiesta di credenziali di sicurezza temporanee nella Guida per l'utente IAM.

ii. Esegui i seguenti comandi per fornire le credenziali al software Core. AWS IoT Greengrass

Linux or Unix

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY
export AWS_SESSION_TOKEN=AQoDYXdzEJr1K...o5OytwEXAMPLE=
```

Windows Command Prompt (CMD)

```
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
set AWS_SESSION_TOKEN=AQoDYXdzEJr1K...o5OytwEXAMPLE=
```

PowerShell

```
$env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
$env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY"
$env:AWS_SESSION_TOKEN="AQoDYXdzEJr1K...o5OytwEXAMPLE="
```

- Utilizza le credenziali a lungo termine di un utente IAM:

  i. Fornisci l'ID della chiave di accesso e la chiave di accesso segreta per il tuo utente IAM. Puoi creare un utente IAM per il provisioning da eliminare successivamente. Per la policy IAM da fornire all'utente, consultaPolicy IAM minima per l'installatore per il provisioning delle risorse. Per ulteriori informazioni su come recuperare le credenziali a lungo termine, consulta Managing access keys for IAM users nella IAM User Guide.

  ii. Esegui i seguenti comandi per fornire le credenziali al AWS IoT Greengrass software Core.

Linux or Unix

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY
```

Windows Command Prompt (CMD)

```
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

PowerShell

```
$env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
$env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY"
```

iii. (Facoltativo) Se hai creato un utente IAM per il provisioning del tuo dispositivo Greengrass, elimina l'utente.

iv. (Facoltativo) Se hai utilizzato l'ID della chiave di accesso e la chiave di accesso segreta di un utente IAM esistente, aggiorna le chiavi dell'utente in modo che non siano più valide. Per ulteriori informazioni, consulta Aggiornamento delle chiavi di accesso nella guida AWS Identity and Access Management per l'utente.

c. In Esegui il programma di installazione, completa i seguenti passaggi.

i. In Scarica il programma di installazione, scegli Copia ed esegui il comando copiato sul tuo dispositivo principale. Questo comando scarica l'ultima versione del software AWS IoT Greengrass Core e la decomprime sul tuo dispositivo.

ii. In Esegui il programma di installazione, scegli Copia ed esegui il comando copiato sul tuo dispositivo principale. Questo comando utilizza i nomi degli AWS IoT oggetti e dei gruppi di oggetti specificati in precedenza per eseguire il programma di installazione del software AWS IoT Greengrass Core e configurare AWS le risorse per il dispositivo principale.

Questo comando esegue inoltre le seguenti operazioni:

- Configura il software AWS IoT Greengrass Core come servizio di sistema che viene eseguito all'avvio. Sui dispositivi Linux, ciò richiede il sistema di inizializzazione Systemd.

  > **Important**
  >
  > Sui dispositivi Windows core, è necessario configurare il software AWS IoT Greengrass Core come servizio di sistema.

- Implementa il componente AWS IoT GreengrassCLI, che è uno strumento da riga di comando che ti consente di sviluppare componenti Greengrass personalizzati sul dispositivo principale.
- Specificare di utilizzare l'utente `ggc_user` del sistema per eseguire componenti software sul dispositivo principale. Sui dispositivi Linux, questo comando specifica anche di utilizzare il gruppo di `ggc_group` sistema e il programma di installazione crea automaticamente l'utente e il gruppo di sistema.

Quando si esegue questo comando, dovrebbero essere visualizzati i seguenti messaggi per indicare che l'installazione è riuscita.

```
Successfully configured Nucleus with provisioned resource details!
Configured Nucleus to deploy aws.greengrass.Cli component
Successfully set up Nucleus as a system service
```

> **Note**
>
> Se avete un dispositivo Linux che non ha systemd, il programma di installazione non configurerà il software come servizio di sistema e non vedrete il messaggio di successo relativo alla configurazione del nucleus come servizio di sistema.

Installazione del software AWS IoT Greengrass Core (CLI)

Per installare e configurare il software AWS IoT Greengrass Core

1. Sul tuo dispositivo principale Greengrass, esegui il seguente comando per passare alla home directory.

   Linux or Unix

   ```
   cd ~
   ```

   Windows Command Prompt (CMD)

   ```
   cd %USERPROFILE%
   ```

   PowerShell

   ```
   cd ~
   ```

2. Sul tuo dispositivo principale, scarica il software AWS IoT Greengrass Core in un file denominato`greengrass-nucleus-latest.zip`.

   Linux or Unix

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
   latest.zip > greengrass-nucleus-latest.zip
   ```

   Windows Command Prompt (CMD)

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
   latest.zip > greengrass-nucleus-latest.zip
   ```

   PowerShell

   ```
   iwr -Uri https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
   latest.zip -OutFile greengrass-nucleus-latest.zip
   ```

   Scaricando questo software accetti l'Accordo di licenza del software Greengrass Core.

3. Decomprimi il software AWS IoT Greengrass Core in una cartella sul dispositivo. Sostituiscilo *GreengrassInstaller*con la cartella che desideri utilizzare.

   Linux or Unix

   ```
   unzip greengrass-nucleus-latest.zip -d GreengrassInstaller && rm greengrass-
   nucleus-latest.zip
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir GreengrassInstaller && tar -xf greengrass-nucleus-latest.zip -
   C GreengrassInstaller && del greengrass-nucleus-latest.zip
   ```

   PowerShell

   ```
   Expand-Archive -Path greengrass-nucleus-latest.zip -DestinationPath .\
   \GreengrassInstaller
   rm greengrass-nucleus-latest.zip
   ```

4. Esegui il comando seguente per avviare il programma di installazione del software AWS IoT Greengrass Core. Questo comando funziona nel modo seguente:

   - Crea le AWS risorse necessarie al funzionamento del dispositivo principale.
   - Configura il software AWS IoT Greengrass Core come servizio di sistema che viene eseguito all'avvio. Sui dispositivi Linux, ciò richiede il sistema di inizializzazione Systemd.

     > **Important**
     >
     > Sui dispositivi Windows core, è necessario configurare il software AWS IoT Greengrass Core come servizio di sistema.

   - Implementa il componente AWS IoT GreengrassCLI, che è uno strumento da riga di comando che ti consente di sviluppare componenti Greengrass personalizzati sul dispositivo principale.
   - Specificare di utilizzare l'utente `ggc_user` del sistema per eseguire componenti software sul dispositivo principale. Sui dispositivi Linux, questo comando specifica anche di utilizzare il gruppo di `ggc_group` sistema e il programma di installazione crea automaticamente l'utente e il gruppo di sistema.

Sostituite i valori degli argomenti nel comando come segue.

a. *`/greengrass/v2`*o *`C:\greengrass\v2`*: il percorso della cartella principale da utilizzare per installare il software AWS IoT Greengrass Core.

b. *`GreengrassInstaller`*. Il percorso della cartella in cui è stato decompresso il programma di installazione del software AWS IoT Greengrass Core.

c. *`regione.`* Il Regione AWS luogo in cui trovare o creare risorse.

d. *`MyGreengrassCore`*. Il nome del AWS IoT dispositivo principale Greengrass. Se l'oggetto non esiste, l'installatore lo crea. Il programma di installazione scarica i certificati per autenticarsi come oggetto. AWS IoT Per ulteriori informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass.

> **Note**
>
> Il nome dell'oggetto non può contenere i due punti (`:`).

e. *`MyGreengrassCoreGroup`*. Il nome del AWS IoT gruppo di oggetti per il dispositivo principale Greengrass. Se il gruppo di oggetti non esiste, l'installatore lo crea e vi aggiunge l'oggetto. Se il gruppo di oggetti esiste e dispone di una distribuzione attiva, il dispositivo principale scarica ed esegue il software specificato dalla distribuzione.

> **Note**
>
> Il nome del gruppo di cose non può contenere i due punti (`:`).

f. *`Greengrass v2 IoT ThingPolicy`*. Il nome della AWS IoT policy che consente ai dispositivi core Greengrass di comunicare con AWS IoT e. AWS IoT Greengrass Se la AWS IoT politica non esiste, il programma di installazione crea una AWS IoT politica permissiva con questo nome. Puoi limitare le autorizzazioni di questa politica in base al tuo caso d'uso. Per ulteriori informazioni, consulta AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali.

g. *`GreenGrass v2 TokenExchangeRole`*. Il nome del ruolo IAM che consente al dispositivo principale Greengrass di ottenere credenziali temporaneeAWS. Se il ruolo non esiste, l'installatore lo crea e crea e allega una policy denominata. *`GreengrassV2TokenExchangeRole`*`Access` Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

h. *GreengrassCoreTokenExchangeRoleAlias*. L'alias del ruolo IAM che consente al dispositivo principale Greengrass di ottenere credenziali temporanee in un secondo momento. Se l'alias del ruolo non esiste, il programma di installazione lo crea e lo indirizza al ruolo IAM specificato. Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

Linux or Unix

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
  -jar ./GreengrassInstaller/lib/Greengrass.jar \
  --aws-region region \
  --thing-name MyGreengrassCore \
  --thing-group-name MyGreengrassCoreGroup \
  --thing-policy-name GreengrassV2IoTThingPolicy \
  --tes-role-name GreengrassV2TokenExchangeRole \
  --tes-role-alias-name GreengrassCoreTokenExchangeRoleAlias \
  --component-default-user ggc_user:ggc_group \
  --provision true \
  --setup-system-service true \
  --deploy-dev-tools true
```

Windows Command Prompt (CMD)

```
java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" ^
  -jar ./GreengrassInstaller/lib/Greengrass.jar ^
  --aws-region region ^
  --thing-name MyGreengrassCore ^
  --thing-group-name MyGreengrassCoreGroup ^
  --thing-policy-name GreengrassV2IoTThingPolicy ^
  --tes-role-name GreengrassV2TokenExchangeRole ^
  --tes-role-alias-name GreengrassCoreTokenExchangeRoleAlias ^
  --component-default-user ggc_user ^
  --provision true ^
  --setup-system-service true ^
  --deploy-dev-tools true
```

PowerShell

```
java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" `
  -jar ./GreengrassInstaller/lib/Greengrass.jar `
```

```
  --aws-region region `
  --thing-name MyGreengrassCore `
  --thing-group-name MyGreengrassCoreGroup `
  --thing-policy-name GreengrassV2IoTThingPolicy `
  --tes-role-name GreengrassV2TokenExchangeRole `
  --tes-role-alias-name GreengrassCoreTokenExchangeRoleAlias `
  --component-default-user ggc_user `
  --provision true `
  --setup-system-service true `
  --deploy-dev-tools true
```

> **Note**
>
> Se utilizzi AWS IoT Greengrass un dispositivo con memoria limitata, puoi controllare la quantità di memoria utilizzata dal software AWS IoT Greengrass Core. Per controllare l'allocazione della memoria, è possibile impostare le opzioni relative alla dimensione dell'heap JVM nel parametro di `jvmOptions` configurazione del componente nucleus. Per ulteriori informazioni, consulta Controlla l'allocazione della memoria con le opzioni JVM.

Quando esegui questo comando, dovresti vedere i seguenti messaggi per indicare che l'installazione è riuscita.

```
Successfully configured Nucleus with provisioned resource details!
Configured Nucleus to deploy aws.greengrass.Cli component
Successfully set up Nucleus as a system service
```

> **Note**
>
> Se avete un dispositivo Linux che non ha systemd, il programma di installazione non configurerà il software come servizio di sistema e non vedrete il messaggio di successo relativo alla configurazione del nucleus come servizio di sistema.

(Facoltativo) Esegui il software Greengrass (Linux)

Se il software è stato installato come servizio di sistema, il programma di installazione lo esegue automaticamente. In caso contrario, è necessario eseguire il software. Per verificare se il programma di installazione ha configurato il software come servizio di sistema, cercate la riga seguente nell'output dell'installatore.

```
Successfully set up Nucleus as a system service
```

Se non vedi questo messaggio, procedi come segue per eseguire il software:

1. Esegui il comando seguente per eseguire il software.

   ```
   sudo /greengrass/v2/alts/current/distro/bin/loader
   ```

   Il software stampa il seguente messaggio se viene avviato correttamente.

   ```
   Launched Nucleus successfully.
   ```

2. È necessario lasciare aperta la shell dei comandi corrente per mantenere in esecuzione il software AWS IoT Greengrass Core. Se usi SSH per connetterti al dispositivo principale, esegui il comando seguente sul tuo computer di sviluppo per aprire una seconda sessione SSH da utilizzare per eseguire comandi aggiuntivi sul dispositivo principale. Sostituisci il nome *utente* con il nome dell'utente a cui accedere e sostituiscilo *pi-ip-address*con l'indirizzo IP del dispositivo.

   ```
   ssh username@pi-ip-address
   ```

Per ulteriori informazioni su come interagire con il servizio di sistema Greengrass, vedere. Configurare il nucleo Greengrass come servizio di sistema

Verifica l'installazione della CLI di Greengrass sul dispositivo

L'implementazione della CLI di Greengrass può richiedere fino a un minuto. Esegui il comando seguente per verificare lo stato della distribuzione. Sostituiscilo *MyGreengrassCore*con il nome del tuo dispositivo principale.

```
aws greengrassv2 list-effective-deployments --core-device-thing-name MyGreengrassCore
```

`coreDeviceExecutionStatus`Indica lo stato della distribuzione sul dispositivo principale. Quando lo stato è`SUCCEEDED`, esegui il comando seguente per verificare che la CLI Greengrass sia installata e funzionante. Sostituisci */greengrass/v2* con il percorso della cartella principale.

Linux or Unix

```
/greengrass/v2/bin/greengrass-cli help
```

Windows Command Prompt (CMD)

```
C:\greengrass\v2\bin\greengrass-cli help
```

PowerShell

```
C:\greengrass\v2\bin\greengrass-cli help
```

Il comando fornisce informazioni di aiuto per la CLI di Greengrass. Se `greengrass-cli` non viene trovato, la distribuzione potrebbe non essere riuscita a installare la CLI di Greengrass. Per ulteriori informazioni, consulta Risoluzione dei problemi AWS IoT Greengrass V2.

Puoi anche eseguire il comando seguente per distribuire manualmente la AWS IoT Greengrass CLI sul tuo dispositivo.

- Sostituisci la *regione* con quella Regione AWS che usi. Assicurati di utilizzare la stessa Regione AWS che hai usato per configurarla AWS CLI sul tuo dispositivo.
- Sostituisci l'*account-id* con il tuo Account AWS ID.
- *MyGreengrassCore*Sostituiscilo con il nome del tuo dispositivo principale.

Linux, macOS, or Unix

```
aws greengrassv2 create-deployment \
  --target-arn "arn:aws:iot:region:account-id:thing/MyGreengrassCore" \
  --components '{
    "aws.greengrass.Cli": {
```

```
      "componentVersion": "2.12.3"
    }
  }'
```

Windows Command Prompt (CMD)

```
aws greengrassv2 create-deployment ^
  --target-arn "arn:aws:iot:region:account-id:thing/MyGreengrassCore" ^
  --components "{\"aws.greengrass.Cli\":{\"componentVersion\":\"2.12.3\"}}"
```

PowerShell

```
aws greengrassv2 create-deployment `
  --target-arn "arn:aws:iot:region:account-id:thing/MyGreengrassCore" `
  --components '{\"aws.greengrass.Cli\":{\"componentVersion\":\"2.12.3\"}}'
```

> **Tip**
>
> È possibile aggiungere */greengrass/v2*/bin (Linux) o *C:\greengrass\v2*\bin (Windows) alla variabile di PATH ambiente per eseguirla `greengrass-cli` senza il percorso assoluto.

Il softwareAWS IoT Greengrass Core e gli strumenti di sviluppo locali vengono eseguiti sul tuo dispositivo. Successivamente, puoi sviluppare unAWS IoT Greengrass componente Hello World sul tuo dispositivo.

Fase 4: Sviluppa e testa un componente sul tuo dispositivo

Un componente è un modulo software che viene eseguito sui dispositivi AWS IoT Greengrass principali. I componenti consentono di creare e gestire applicazioni complesse come elementi costitutivi discreti che puoi riutilizzare da un dispositivo core Greengrass a un altro. Ogni componente è composto da una ricetta e da artefatti.

- Ricette

  Ogni componente contiene un file di ricette, che ne definisce i metadati. La ricetta specifica anche i parametri di configurazione del componente, le dipendenze dei componenti, il ciclo di

vita e la compatibilità della piattaforma. Il ciclo di vita del componente definisce i comandi che installano, eseguono e spengono il componente. Per ulteriori informazioni, consulta AWS IoT Greengrassriferimento alla ricetta del componente.

È possibile definire ricette in formato JSON o YAML.

- Artefatti

  I componenti possono avere un numero qualsiasi di artefatti, che sono componenti binari. Gli artefatti possono includere script, codice compilato, risorse statiche e qualsiasi altro file utilizzato da un componente. I componenti possono anche consumare artefatti derivanti dalle dipendenze dei componenti.

Con AWS IoT Greengrass, puoi utilizzare la Greengrass CLI per sviluppare e testare i componenti localmente su un dispositivo core Greengrass senza interazione con il Cloud. AWS Una volta completato il componente locale, puoi utilizzare la ricetta e gli artefatti del componente per creare quel componente nel AWS IoT Greengrass servizio nel AWS Cloud e quindi distribuirlo su tutti i tuoi dispositivi principali Greengrass. Per ulteriori informazioni sui componenti, vedere. Sviluppa AWS IoT Greengrass componenti

In questa sezione, imparerai come creare ed eseguire un componente Hello World di base localmente sul tuo dispositivo principale.

Per sviluppare un componente Hello World sul tuo dispositivo

1. Crea una cartella per i componenti con sottocartelle per ricette e artefatti. Esegui i seguenti comandi sul tuo dispositivo principale Greengrass per creare queste cartelle e passare alla cartella dei componenti. Sostituisci *~/greengrassv2* o *%USERPROFILE%\ greengrassv2* con il percorso della cartella da utilizzare per lo sviluppo locale.

   Linux or Unix

   ```
   mkdir -p ~/greengrassv2/{recipes,artifacts}
   cd ~/greengrassv2
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir %USERPROFILE%\greengrassv2\\recipes, %USERPROFILE%\greengrassv2\\artifacts
   cd %USERPROFILE%\greengrassv2
   ```

PowerShell

```
mkdir ~/greengrassv2/recipes, ~/greengrassv2/artifacts
cd ~/greengrassv2
```

2. Usa un editor di testo per creare un file di ricette che definisca i metadati, i parametri, le dipendenze, il ciclo di vita e le funzionalità della piattaforma del componente. Includi la versione del componente nel nome del file di ricetta in modo da poter identificare quale ricetta riflette quale versione del componente. Puoi scegliere il formato YAML o JSON per la tua ricetta.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

   JSON

   ```
   nano recipes/com.example.HelloWorld-1.0.0.json
   ```

   YAML

   ```
   nano recipes/com.example.HelloWorld-1.0.0.yaml
   ```

   > **Note**
   >
   > AWS IoT Greengrass utilizza versioni semantiche per i componenti. Le versioni semantiche seguono una delle principali. minore. sistema di numerazione delle patch. Ad esempio, la versione `1.0.0` rappresenta la prima release principale di un componente. Per ulteriori informazioni, consultate la specifica della versione semantica.

3. Incolla la seguente ricetta nel file.

   JSON

   ```
   {
     "RecipeFormatVersion": "2020-01-25",
     "ComponentName": "com.example.HelloWorld",
     "ComponentVersion": "1.0.0",
     "ComponentDescription": "My first AWS IoT Greengrass component.",
     "ComponentPublisher": "Amazon",
   ```

```
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "Message": "world"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "run": "python3 -u {artifacts:path}/hello_world.py {configuration:/
Message}"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "run": "py -3 -u {artifacts:path}/hello_world.py {configuration:/
Message}"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.HelloWorld
ComponentVersion: '1.0.0'
ComponentDescription: My first AWS IoT Greengrass component.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    Message: world
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      run: |
```

```
        python3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
  - Platform:
      os: windows
    Lifecycle:
      run: |
        py -3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
```

La `ComponentConfiguration` sezione di questa ricetta definisce un parametro`Message`, il cui valore predefinito è. `world` La `Manifests` sezione definisce un manifesto, che è un insieme di istruzioni e artefatti del ciclo di vita per una piattaforma. È possibile definire più manifesti per specificare istruzioni di installazione diverse per varie piattaforme, ad esempio. Nel manifesto, la `Lifecycle` sezione indica al dispositivo principale Greengrass di eseguire lo script Hello World con il valore `Message` del parametro come argomento.

4. Eseguite il comando seguente per creare una cartella per gli artefatti del componente.

   Linux or Unix

   ```
   mkdir -p artifacts/com.example.HelloWorld/1.0.0
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir artifacts\com.example.HelloWorld\1.0.0
   ```

   PowerShell

   ```
   mkdir artifacts\com.example.HelloWorld\1.0.0
   ```

   Important

   È necessario utilizzare il seguente formato per il percorso della cartella degli artefatti. Includete il nome e la versione del componente specificati nella ricetta.

   ```
   artifacts/componentName/componentVersion/
   ```

5. Usa un editor di testo per creare un file artefatto di script Python per il tuo componente Hello World.

Ad esempio, su un sistema basato su Linux, puoi eseguire il comando seguente per usare GNU nano per creare il file.

```
nano artifacts/com.example.HelloWorld/1.0.0/hello_world.py
```

Copia e incolla il seguente script Python nel file.

```
import sys

message = "Hello, %s!" % sys.argv[1]

# Print the message to stdout, which Greengrass saves in a log file.
print(message)
```

6. Utilizza la AWS IoT Greengrass CLI locale per gestire i componenti sul tuo dispositivo principale Greengrass.

   Esegui il comando seguente per distribuire il componente nel core. AWS IoT Greengrass Sostituisci */greengrass/v2* o *C:\greengrass\v2* con la cartella AWS IoT Greengrass V2 principale e sostituisci *~/greengrassv2 o %USERPROFILE%\ greengrassv2* con la cartella di sviluppo dei componenti.

   Linux or Unix

   ```
   sudo /greengrass/v2/bin/greengrass-cli deployment create \
     --recipeDir ~/greengrassv2/recipes \
     --artifactDir ~/greengrassv2/artifacts \
     --merge "com.example.HelloWorld=1.0.0"
   ```

   Windows Command Prompt (CMD)

   ```
   C:\greengrass\v2\bin\greengrass-cli deployment create ^
     --recipeDir %USERPROFILE%\greengrassv2\recipes ^
     --artifactDir %USERPROFILE%\greengrassv2\artifacts ^
     --merge "com.example.HelloWorld=1.0.0"
   ```

   PowerShell

   ```
   C:\greengrass\v2\bin\greengrass-cli deployment create `
   ```

```
  --recipeDir ~/greengrassv2/recipes `
  --artifactDir ~/greengrassv2/artifacts `
  --merge "com.example.HelloWorld=1.0.0"
```

Questo comando aggiunge il componente che utilizza la ricetta in `recipes` e lo script Python in. `artifacts` L'`--merge`opzione aggiunge o aggiorna il componente e la versione specificati.

7. Il software AWS IoT Greengrass Core salva lo stdout dal processo componente nei file di registro nella `logs` cartella. Esegui il comando seguente per verificare che il componente Hello World venga eseguito e stampi i messaggi.

   Linux or Unix

   ```
   sudo tail -f /greengrass/v2/logs/com.example.HelloWorld.log
   ```

   Windows Command Prompt (CMD)

   ```
   type C:\greengrass\v2\logs\com.example.HelloWorld.log
   ```

   Il `type` comando scrive il contenuto del file nel terminale. Esegui questo comando più volte per osservare le modifiche nel file.

   PowerShell

   ```
   gc C:\greengrass\v2\logs\com.example.HelloWorld.log -Tail 10 -Wait
   ```

   Dovrebbero essere visualizzati messaggi simili a quelli dell'esempio seguente.

   ```
   Hello, world!
   ```

   > **Note**
   >
   > Se il file non esiste, la distribuzione locale potrebbe non essere ancora completa. Se il file non esiste entro 15 secondi, è probabile che la distribuzione non sia riuscita. Ciò può verificarsi, ad esempio, se la ricetta non è valida. Esegui il comando seguente per visualizzare il file di registro AWS IoT Greengrass principale. Questo file include i log del servizio di distribuzione del dispositivo principale Greengrass.

Linux or Unix

```
sudo tail -f /greengrass/v2/logs/greengrass.log
```

Windows Command Prompt (CMD)

```
type C:\greengrass\v2\logs\greengrass.log
```

Il `type` comando scrive il contenuto del file nel terminale. Esegui questo comando più volte per osservare le modifiche nel file.

PowerShell

```
gc C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
```

8. Modifica il componente locale per iterare e testare il codice. Apri `hello_world.py` in un editor di testo e aggiungi il codice seguente alla riga 4 per modificare il messaggio registrato dal AWS IoT Greengrass core.

```
message += " Greetings from your first Greengrass component."
```

Lo `hello_world.py` script dovrebbe ora avere i seguenti contenuti.

```
import sys

message = "Hello, %s!" % sys.argv[1]
message += " Greetings from your first Greengrass component."

# Print the message to stdout, which Greengrass saves in a log file.
print(message)
```

9. Esegui il comando seguente per aggiornare il componente con le tue modifiche.

Linux or Unix

```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
  --recipeDir ~/greengrassv2/recipes \
  --artifactDir ~/greengrassv2/artifacts \
  --merge "com.example.HelloWorld=1.0.0"
```

Windows Command Prompt (CMD)

```
C:\greengrass\v2\bin\greengrass-cli deployment create ^
  --recipeDir %USERPROFILE%\greengrassv2\recipes ^
  --artifactDir %USERPROFILE%\greengrassv2\artifacts ^
  --merge "com.example.HelloWorld=1.0.0"
```

PowerShell

```
C:\greengrass\v2\bin\greengrass-cli deployment create `
  --recipeDir ~/greengrassv2/recipes `
  --artifactDir ~/greengrassv2/artifacts `
  --merge "com.example.HelloWorld=1.0.0"
```

Questo comando aggiorna il `com.example.HelloWorld` componente con l'ultimo artefatto Hello World.

10. Eseguite il comando seguente per riavviare il componente. Quando riavviate un componente, il dispositivo principale utilizza le ultime modifiche.

    Linux or Unix

    ```
    sudo /greengrass/v2/bin/greengrass-cli component restart \
      --names "com.example.HelloWorld"
    ```

    Windows Command Prompt (CMD)

    ```
    C:\greengrass\v2\bin\greengrass-cli component restart ^
      --names "com.example.HelloWorld"
    ```

    PowerShell

    ```
    C:\greengrass\v2\bin\greengrass-cli component restart `
      --names "com.example.HelloWorld"
    ```

11. Controlla nuovamente il registro per verificare che il componente Hello World stampi il nuovo messaggio.

Linux or Unix

```
sudo tail -f /greengrass/v2/logs/com.example.HelloWorld.log
```

Windows Command Prompt (CMD)

```
type C:\greengrass\v2\logs\com.example.HelloWorld.log
```

Il `type` comando scrive il contenuto del file nel terminale. Esegui questo comando più volte per osservare le modifiche nel file.

PowerShell

```
gc C:\greengrass\v2\logs\com.example.HelloWorld.log -Tail 10 -Wait
```

Dovrebbero essere visualizzati messaggi simili a quelli dell'esempio seguente.

```
Hello, world! Greetings from your first Greengrass component.
```

12. È possibile aggiornare i parametri di configurazione del componente per testare diverse configurazioni. Quando si distribuisce un componente, è possibile specificare un aggiornamento della configurazione, che definisce come modificare la configurazione del componente sul dispositivo principale. È possibile specificare quali valori di configurazione ripristinare ai valori predefiniti e i nuovi valori di configurazione da unire al dispositivo principale. Per ulteriori informazioni, consulta Aggiornamento delle configurazioni dei componenti.

    Esegui questa operazione:

    a. Utilizzate un editor di testo per creare un file chiamato `hello-world-config-update.json` a contenere l'aggiornamento della configurazione

       Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

       ```
       nano hello-world-config-update.json
       ```

    b. Copiate e incollate il seguente oggetto JSON nel file. Questo oggetto JSON definisce un aggiornamento della configurazione che unisce il valore al `Message` parametro `friend`

per aggiornarne il valore. Questo aggiornamento della configurazione non specifica alcun valore da reimpostare. Non è necessario reimpostare il `Message` parametro perché l'aggiornamento di fusione sostituisce il valore esistente.

```
{
  "com.example.HelloWorld": {
    "MERGE": {
      "Message": "friend"
    }
  }
}
```

c. Esegui il comando seguente per distribuire l'aggiornamento della configurazione nel componente Hello World.

Linux or Unix

```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
  --merge "com.example.HelloWorld=1.0.0" \
  --update-config hello-world-config-update.json
```

Windows Command Prompt (CMD)

```
C:\greengrass\v2\bin\greengrass-cli deployment create ^
  --merge "com.example.HelloWorld=1.0.0" ^
  --update-config hello-world-config-update.json
```

PowerShell

```
C:\greengrass\v2\bin\greengrass-cli deployment create `
  --merge "com.example.HelloWorld=1.0.0" `
  --update-config hello-world-config-update.json
```

d. Controlla nuovamente il registro per verificare che il componente Hello World emetta il nuovo messaggio.

Linux or Unix

```
sudo tail -f /greengrass/v2/logs/com.example.HelloWorld.log
```

Windows Command Prompt (CMD)

```
type C:\greengrass\v2\logs\com.example.HelloWorld.log
```

Il `type` comando scrive il contenuto del file nel terminale. Esegui questo comando più volte per osservare le modifiche nel file.

PowerShell

```
gc C:\greengrass\v2\logs\com.example.HelloWorld.log -Tail 10 -Wait
```

Dovrebbero essere visualizzati messaggi simili a quelli dell'esempio seguente.

```
Hello, friend! Greetings from your first Greengrass component.
```

13. Dopo aver terminato il test del componente, rimuovilo dal dispositivo principale. Esegui il comando seguente.

    Linux or Unix

    ```
    sudo /greengrass/v2/bin/greengrass-cli deployment create --
    remove="com.example.HelloWorld"
    ```

    Windows Command Prompt (CMD)

    ```
    C:\greengrass\v2\bin\greengrass-cli deployment create --
    remove="com.example.HelloWorld"
    ```

    PowerShell

    ```
    C:\greengrass\v2\bin\greengrass-cli deployment create --
    remove="com.example.HelloWorld"
    ```

    Important

    Questo passaggio è necessario per ridistribuire il componente sul dispositivo principale dopo averlo caricato su. AWS IoT Greengrass In caso contrario, la distribuzione fallisce

con un errore di compatibilità della versione perché la distribuzione locale specifica una versione diversa del componente.

Esegui il comando seguente e verifica che il `com.example.HelloWorld` componente non compaia nell'elenco dei componenti del dispositivo.

Linux or Unix

```
sudo /greengrass/v2/bin/greengrass-cli component list
```

Windows Command Prompt (CMD)

```
C:\greengrass\v2\bin\greengrass-cli component list
```

PowerShell

```
C:\greengrass\v2\bin\greengrass-cli component list
```

Il componente Hello World è completo e ora puoi caricarlo sul servizio AWS IoT Greengrass cloud. Quindi, puoi distribuire il componente sui dispositivi principali Greengrass.

Fase 5: Crea il tuo componente nel AWS IoT Greengrass servizio

Quando finisci di sviluppare un componente sul tuo dispositivo principale, puoi caricarlo sul AWS IoT Greengrass servizio inCloud AWS. Puoi anche creare direttamente il componente nella AWS IoT Greengrassconsole. AWS IoT Greengrassfornisce un servizio di gestione dei componenti che ospita i componenti in modo da poterli distribuire su singoli dispositivi o flotte di dispositivi. Per caricare un componente nel AWS IoT Greengrass servizio, è necessario completare i seguenti passaggi:

- Carica gli artefatti dei componenti in un bucket S3.
- Aggiungi l'URI Amazon Simple Storage Service (Amazon S3) di ogni elemento alla ricetta del componente.
- Crea un componente AWS IoT Greengrass dalla ricetta del componente.

In questa sezione, completa questi passaggi sul tuo dispositivo principale Greengrass per caricare il componente Hello World sul AWS IoT Greengrass servizio.

Crea il tuo componente in AWS IoT Greengrass (console)

1. Usa un bucket S3 nel tuo AWS account per ospitare gli artefatti AWS IoT Greengrass dei componenti. Quando distribuisci il componente su un dispositivo principale, il dispositivo scarica gli artefatti del componente dal bucket.

   Puoi usare un bucket S3 esistente oppure puoi crearne uno nuovo.

   a. Nella console Amazon S3, in Bucket, scegli Crea bucket.

   b. Per Bucket name, inserisci un nome di bucket univoco. Per esempio, è possibile utilizzare **`greengrass-component-artifacts-`*`region`*`-`*`123456789012`***. Sostituisci *`123456789012`* con l'ID del tuo AWS account e la *`regione`* con Regione AWS quella che usi per questo tutorial.

   c. Per la AWSregione, seleziona la AWS regione che usi per questo tutorial.

   d. Seleziona Crea bucket.

   e. In Bucket, scegli il bucket che hai creato, carica `hello_world.py` lo script nella `artifacts/com.example.HelloWorld/1.0.0` cartella all'interno del bucket. Per informazioni sul caricamento di oggetti nei bucket S3, consulta Caricamento di oggetti nella Amazon Simple Storage Service User Guide.

   f. Copia l'URI S3 dell'`hello_world.py`oggetto nel bucket S3. Questo URI dovrebbe essere simile all'esempio seguente. Sostituisci *`DOC-EXAMPLE-BUCKET con il nome del bucket S3`*.

   ```
   s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.HelloWorld/1.0.0/hello_world.py
   ```

2. Consenti al dispositivo principale di accedere agli artefatti dei componenti nel bucket S3.

   Ogni dispositivo principale ha un ruolo IAM del dispositivo principale che gli consente di interagire AWS IoT e inviare log al cloud. AWS Per impostazione predefinita, questo ruolo del dispositivo non consente l'accesso ai bucket S3, quindi è necessario creare e allegare una policy che consenta al dispositivo principale di recuperare gli artefatti dei componenti dal bucket S3.

   Se il ruolo del tuo dispositivo consente già l'accesso al bucket S3, puoi saltare questo passaggio. Altrimenti, crea una policy IAM che consenta l'accesso e collegala al ruolo, come segue:

a. Nel menu di navigazione della console IAM, scegli Policies, quindi scegli Crea policy.

b. Nella scheda JSON, sostituire il contenuto del segnaposto con la seguente policy. Sostituisci *DOC-EXAMPLE-BUCKET con il nome del bucket* S3 che contiene gli artefatti dei componenti per il dispositivo principale da scaricare.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
    }
  ]
}
```

c. Seleziona Avanti.

d. Nella sezione Dettagli della politica, per Nome, inserisci. **MyGreengrassV2ComponentArtifactPolicy**

e. Scegli Crea policy.

f. Nel menu di navigazione della console IAM, scegli Ruolo, quindi scegli il nome del ruolo per il dispositivo principale. Hai specificato questo nome di ruolo quando hai installato il software AWS IoT Greengrass Core. Se non hai specificato un nome, l'impostazione predefinita è`GreengrassV2TokenExchangeRole`.

g. In Autorizzazioni, scegli Aggiungi autorizzazioni, quindi scegli Allega politiche.

h. Nella pagina Aggiungi autorizzazioni, seleziona la casella di controllo accanto alla **MyGreengrassV2ComponentArtifactPolicy** politica che hai creato, quindi scegli Aggiungi autorizzazioni.

3. Usa la ricetta del componente per creare un componente nella AWS IoT Greengrass console.

a. Nel menu di navigazione della AWS IoT Greengrassconsole, scegli Componenti, quindi scegli Crea componente.

b. In Informazioni sui componenti, scegli Inserisci la ricetta come JSON. La ricetta segnaposto dovrebbe essere simile all'esempio seguente.

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.HelloWorld",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "My first AWS IoT Greengrass component.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "Message": "world"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "run": "python3 -u {artifacts:path}/hello_world.py \"{configuration:/
Message}\""
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.HelloWorld/1.0.0/hello_world.py"
        }
      ]
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "run": "py -3 -u {artifacts:path}/hello_world.py \"{configuration:/
Message}\""
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.HelloWorld/1.0.0/hello_world.py"
        }
      ]
    }
  ]
```

```
}
```

c. Sostituisci l'URI segnaposto in ogni `Artifacts` sezione con l'URI S3 del tuo oggetto. `hello_world.py`

d. Scegli Crea componente.

e. Sul com.example. HelloWorldpagina del componente, verifica che lo stato del componente sia Implementabile.

Crea il tuo componente in AWS IoT Greengrass () AWS CLI

Per caricare il tuo componente Hello World

1. Usa un bucket S3 Account AWS per ospitare gli artefatti AWS IoT Greengrass dei componenti. Quando distribuisci il componente su un dispositivo principale, il dispositivo scarica gli artefatti del componente dal bucket.

   Puoi utilizzare un bucket S3 esistente o eseguire il seguente comando per creare un bucket. Questo comando crea un bucket con il tuo Account AWS ID e Regione AWS per formare un nome di bucket univoco. Sostituisci *123456789012* con il tuo Account AWS ID e la tua *regione* con quelli Regione AWS che usi per questo tutorial.

   ```
   aws s3 mb s3://greengrass-component-artifacts-123456789012-region
   ```

   Il comando restituisce le seguenti informazioni se la richiesta ha esito positivo.

   ```
   make_bucket: greengrass-component-artifacts-123456789012-region
   ```

2. Consenti al dispositivo principale di accedere agli artefatti dei componenti nel bucket S3.

   Ogni dispositivo principale ha un ruolo IAM del dispositivo principale che gli consente di interagire AWS IoT e inviare log a. Cloud AWS Per impostazione predefinita, questo ruolo del dispositivo non consente l'accesso ai bucket S3, quindi è necessario creare e allegare una policy che consenta al dispositivo principale di recuperare gli artefatti dei componenti dal bucket S3.

   Se il ruolo del dispositivo principale consente già l'accesso al bucket S3, puoi saltare questo passaggio. Altrimenti, crea una policy IAM che consenta l'accesso e collegala al ruolo, come segue:

a. Crea un file chiamato `component-artifact-policy.json` e copia il seguente codice JSON nel file. Questa politica consente l'accesso a tutti i file in un bucket S3. Sostituisci *DOC-EXAMPLE-BUCKET con il nome del bucket S3.*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
    }
  ]
}
```

b. Esegui il comando seguente per creare la politica dal documento di policy in. `component-artifact-policy.json`

Linux or Unix

```
aws iam create-policy \\
  --policy-name MyGreengrassV2ComponentArtifactPolicy \\
  --policy-document file://component-artifact-policy.json
```

Windows Command Prompt (CMD)

```
aws iam create-policy ^
  --policy-name MyGreengrassV2ComponentArtifactPolicy ^
  --policy-document file://component-artifact-policy.json
```

PowerShell

```
aws iam create-policy `
  --policy-name MyGreengrassV2ComponentArtifactPolicy `
  --policy-document file://component-artifact-policy.json
```

Copia la policy Amazon Resource Name (ARN) dai metadati della policy nell'output. Utilizzerai questo ARN per collegare questa policy al ruolo principale del dispositivo nel passaggio successivo.

c. Esegui il comando seguente per collegare la policy al ruolo principale del dispositivo. Sostituisci *GreenGrassV2 TokenExchangeRole* con il nome del ruolo per il dispositivo principale. Hai specificato questo nome di ruolo quando hai installato il AWS IoT Greengrass software Core. Sostituisci l'ARN della policy con l'ARN del passaggio precedente.

Linux or Unix

```
aws iam attach-role-policy \\
  --role-name GreengrassV2TokenExchangeRole \\
  --policy-arn
 arn:aws:iam::123456789012:policy/MyGreengrassV2ComponentArtifactPolicy
```

Windows Command Prompt (CMD)

```
aws iam attach-role-policy ^
  --role-name GreengrassV2TokenExchangeRole ^
  --policy-arn
 arn:aws:iam::123456789012:policy/MyGreengrassV2ComponentArtifactPolicy
```

PowerShell

```
aws iam attach-role-policy `
  --role-name GreengrassV2TokenExchangeRole `
  --policy-arn
 arn:aws:iam::123456789012:policy/MyGreengrassV2ComponentArtifactPolicy
```

Se il comando non ha alcun risultato, è riuscito. Il dispositivo principale può ora accedere agli artefatti caricati in questo bucket S3.

3. Carica l'elemento dello script Hello World Python nel bucket S3.

Esegui il comando seguente per caricare lo script nello stesso percorso nel bucket in cui lo script si trova nel core. AWS IoT Greengrass Sostituisci *DOC-EXAMPLE-BUCKET con il nome del bucket S3*.

Linux or Unix

```
aws s3 cp \
  artifacts/com.example.HelloWorld/1.0.0/hello_world.py \
  s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.HelloWorld/1.0.0/hello_world.py
```

Windows Command Prompt (CMD)

```
aws s3 cp ^
  artifacts/com.example.HelloWorld/1.0.0/hello_world.py ^
  s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.HelloWorld/1.0.0/hello_world.py
```

PowerShell

```
aws s3 cp `
  artifacts/com.example.HelloWorld/1.0.0/hello_world.py `
  s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.HelloWorld/1.0.0/hello_world.py
```

Il comando emette una riga che inizia con se la richiesta ha esito positivo. `upload:`

4. Aggiungi l'URI Amazon S3 dell'artefatto alla ricetta del componente.

   L'URI Amazon S3 è composto dal nome del bucket e dal percorso dell'oggetto artefatto nel bucket. L'URI Amazon S3 del tuo elemento di script è l'URI in cui carichi l'artefatto nel passaggio precedente. Questo URI dovrebbe essere simile all'esempio seguente. Sostituisci *DOC-EXAMPLE-BUCKET con il nome del bucket S3*.

   ```
   s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.HelloWorld/1.0.0/hello_world.py
   ```

   Per aggiungere l'artefatto alla ricetta, aggiungi un elenco `Artifacts` contenente una struttura con l'URI di Amazon S3.

   JSON

   ```
   "Artifacts": [
     {
       "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.HelloWorld/1.0.0/
   hello_world.py"
     }
   ```

```
]
```

Apri il file della ricetta in un editor di testo.

Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

```
nano recipes/com.example.HelloWorld-1.0.0.json
```

Aggiungi l'artefatto alla ricetta. Il file della ricetta dovrebbe essere simile all'esempio seguente.

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.HelloWorld",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "My first AWS IoT Greengrass component.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "Message": "world"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "run": "python3 -u {artifacts:path}/hello_world.py \"{configuration:/
Message}\""
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.HelloWorld/1.0.0/hello_world.py"
        }
      ]
    },
    {
      "Platform": {
        "os": "windows"
```

```
      },
      "Lifecycle": {
        "run": "py -3 -u {artifacts:path}/hello_world.py \"{configuration:/
Message}\""
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.HelloWorld/1.0.0/hello_world.py"
        }
      ]
    }
  ]
}
```

YAML

```
Artifacts:
  - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.HelloWorld/1.0.0/
hello_world.py
```

Aprite il file delle ricette in un editor di testo.

Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

```
nano recipes/com.example.HelloWorld-1.0.0.yaml
```

Aggiungi l'artefatto alla ricetta. Il file della ricetta dovrebbe essere simile all'esempio seguente.

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.HelloWorld
ComponentVersion: '1.0.0'
ComponentDescription: My first AWS IoT Greengrass component.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    Message: world
Manifests:
```

```
  - Platform:
      os: linux
    Lifecycle:
      run: |
        python3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.HelloWorld/1.0.0/
hello_world.py
  - Platform:
      os: windows
    Lifecycle:
      run: |
        py -3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.HelloWorld/1.0.0/
hello_world.py
```

5. Crea una risorsa componente AWS IoT Greengrass dalla ricetta. Eseguite il comando seguente per creare il componente dalla ricetta, che fornite come file binario.

   JSON

   ```
   aws greengrassv2 create-component-version --inline-recipe fileb://recipes/
   com.example.HelloWorld-1.0.0.json
   ```

   YAML

   ```
   aws greengrassv2 create-component-version --inline-recipe fileb://recipes/
   com.example.HelloWorld-1.0.0.yaml
   ```

   La risposta è simile all'esempio seguente se la richiesta ha esito positivo.

   ```
   {
     "arn":
    "arn:aws:greengrass:region:123456789012:components:com.example.HelloWorld:versions:1.0.0",
     "componentName": "com.example.HelloWorld",
     "componentVersion": "1.0.0",
     "creationTimestamp": "Mon Nov 30 09:04:05 UTC 2020",
     "status": {
       "componentState": "REQUESTED",
       "message": "NONE",
   ```

```
    "errors": {}
  }
}
```

Copia il file `arn` dall'output per verificare lo stato del componente nel passaggio successivo.

> **Note**
>
> Puoi anche vedere il tuo componente Hello World nella AWS IoT Greengrassconsole nella pagina Componenti.

6. Verifica che il componente sia stato creato e sia pronto per essere distribuito. Quando create un componente, il suo stato è`REQUESTED`. Quindi, AWS IoT Greengrass verifica che il componente sia implementabile. È possibile eseguire il comando seguente per interrogare lo stato del componente e verificare che il componente sia distribuibile. Sostituisci `arn` con l'ARN del passaggio precedente.

```
aws greengrassv2 describe-component --arn
 "arn:aws:greengrass:region:123456789012:components:com.example.HelloWorld:versions:1.0.0"
```

Se il componente viene convalidato, la risposta indica che lo stato del componente è. `DEPLOYABLE`

```
{
  "arn":
 "arn:aws:greengrass:region:123456789012:components:com.example.HelloWorld:versions:1.0.0",
  "componentName": "com.example.HelloWorld",
  "componentVersion": "1.0.0",
  "creationTimestamp": "2020-11-30T18:04:05.823Z",
  "publisher": "Amazon",
  "description": "My first Greengrass component.",
  "status": {
    "componentState": "DEPLOYABLE",
    "message": "NONE",
    "errors": {}
  },
  "platforms": [
    {
      "os": "linux",
      "architecture": "all"
```

```
    }
  ]
}
```

Il tuo componente Hello World è ora disponibile inAWS IoT Greengrass. Puoi reinstallarlo su questo dispositivo principale Greengrass o su altri dispositivi principali.

Fase 6: Implementazione del componente

ConAWS IoT Greengrass, puoi distribuire componenti su singoli dispositivi o gruppi di dispositivi. Quando distribuisci un componente, AWS IoT Greengrass installa ed esegue il software di quel componente su ogni dispositivo di destinazione. È necessario specificare i componenti da distribuire e l'aggiornamento della configurazione da distribuire per ciascun componente. È inoltre possibile controllare il modo in cui la distribuzione viene distribuita ai dispositivi destinatari della distribuzione. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.

In questa sezione, ridistribuisci il componente Hello World sul tuo dispositivo principale Greengrass.

Implementa il tuo componente (console)

1. Nel menu di navigazione AWS IoT Greengrassdella console, scegli Componenti.
2. Nella pagina Componenti, nella scheda I miei componenti, scegli com.example.HelloWorld.
3. Nella pagina com.example.HelloWorld, scegli (Distribuisci).
4. Da Aggiungi alla distribuzione, scegli Crea nuova distribuzione, quindi scegli Avanti.
5. Nella pagina Specifica destinazione, procedi come segue:
   a. Nella casella Name (Nome), inserisci **`Deployment for MyGreengrassCore`**.
   b. Per Deployment target, scegli Core device e il nome dell'AWS IoToggetto per il tuo dispositivo principale. Il valore predefinito in questo tutorial è *`MyGreengrassCore`*.
   c. Seleziona Avanti.
6. Nella pagina Seleziona componenti, in I miei componenti, verifica che il com.example.HelloWorldcomponente sia selezionato e scegli Avanti.
7. Nella pagina Configura componenti com.example.HelloWorld, scegli ed esegui le seguenti operazioni:

a. Scegli Configura componente.

b. In Aggiornamento della configurazione, in Configurazione da unire, inserisci la seguente configurazione.

```
{
  "Message": "universe"
}
```

Questo aggiornamento della configurazione imposta il `Message` parametro Hello World `universe` per il dispositivo in questa distribuzione.

c. Scegli Conferma.

d. Seleziona Avanti.

8. Nella pagina Configura impostazioni avanzate, mantieni le impostazioni di configurazione predefinite e scegli Avanti.

9. Nella pagina Review (Verifica), scegli Deploy (Distribuisci).

10. Verifica che la distribuzione sia completata correttamente. La distribuzione può richiedere alcuni minuti. Controlla il registro di Hello World per verificare la modifica. Esegui il seguente comando sul tuo dispositivo principale Greengrass.

Linux or Unix

```
sudo tail -f /greengrass/v2/logs/com.example.HelloWorld.log
```

Windows Command Prompt (CMD)

```
type C:\greengrass\v2\\logs\\com.example.HelloWorld.log
```

PowerShell

```
gc C:\greengrass\v2\\logs\\com.example.HelloWorld.log -Tail 10 -Wait
```

Dovresti vedere messaggi simili all'esempio seguente.

```
Hello, universe! Greetings from your first Greengrass component.
```

> **Note**
>
> Se i messaggi di registro non vengono modificati, la distribuzione non è riuscita o non ha raggiunto il dispositivo principale. Ciò può verificarsi se il dispositivo principale non è connesso a Internet o non dispone delle autorizzazioni per recuperare artefatti dal bucket S3. Esegui il seguente comando sul tuo dispositivo principale per visualizzare il file di registro del software Core. AWS IoT Greengrass Questo file include i log del servizio di distribuzione del dispositivo principale Greengrass.
>
> Linux or Unix
>
> ```
> sudo tail -f /greengrass/v2/logs/greengrass.log
> ```
>
> Windows Command Prompt (CMD)
>
> ```
> type C:\greengrass\v2\\logs\\greengrass.log
> ```
>
> Il `type` comando scrive il contenuto del file nel terminale. Esegui questo comando più volte per osservare le modifiche nel file.
>
> PowerShell
>
> ```
> gc C:\greengrass\v2\\logs\\greengrass.log -Tail 10 -Wait
> ```
>
> Per ulteriori informazioni, consulta Risoluzione dei problemi AWS IoT Greengrass V2.

Distribuisci il tuo componente () AWS CLI

Per distribuire il componente Hello World

1. Sul tuo computer di sviluppo, crea un file chiamato `hello-world-deployment.json` e copia il seguente codice JSON nel file. Questo file definisce i componenti e le configurazioni da distribuire.

```
{
  "components": {
```

```
    "com.example.HelloWorld": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"Message\":\"universe\"}"
      }
    }
  }
}
```

Questo file di configurazione specifica di distribuire la versione `1.0.0` del componente Hello World sviluppata e pubblicata nella procedura precedente. `configurationUpdate`Specificano di unire la configurazione del componente in una stringa con codifica JSON. Questo aggiornamento della configurazione imposta il `Message` parametro Hello World sul dispositivo in `universe` questa distribuzione.

2. Esegui il seguente comando per distribuire il componente sul tuo dispositivo principale Greengrass. È possibile eseguire la distribuzione su oggetti, che sono dispositivi singoli, o gruppi di oggetti, che sono gruppi di dispositivi. Sostituiscilo *`MyGreengrassCore`*con il AWS IoT nome del dispositivo principale.

   Linux or Unix

   ```
   aws greengrassv2 create-deployment \
     --target-arn "arn:aws:iot:region:account-id:thing/MyGreengrassCore" \
     --cli-input-json file://hello-world-deployment.json
   ```

   Windows Command Prompt (CMD)

   ```
   aws greengrassv2 create-deployment ^
     --target-arn "arn:aws:iot:region:account-id:thing/MyGreengrassCore" ^
     --cli-input-json file://hello-world-deployment.json
   ```

   PowerShell

   ```
   aws greengrassv2 create-deployment `
     --target-arn "arn:aws:iot:region:account-id:thing/MyGreengrassCore" `
     --cli-input-json file://hello-world-deployment.json
   ```

   Il comando restituisce una risposta simile all'esempio seguente.

```
{
  "deploymentId": "deb69c37-314a-4369-a6a1-3dff9fce73a9",
  "iotJobId": "b5d92151-6348-4941-8603-bdbfb3e02b75",
  "iotJobArn": "arn:aws:iot:region:account-id:job/b5d92151-6348-4941-8603-
bdbfb3e02b75"
}
```

3. Verificare che la distribuzione venga completata correttamente. La distribuzione può richiedere alcuni minuti. Controlla il registro di Hello World per verificare la modifica. Esegui il seguente comando sul tuo dispositivo principale Greengrass.

   Linux or Unix

   ```
   sudo tail -f /greengrass/v2/logs/com.example.HelloWorld.log
   ```

   Windows Command Prompt (CMD)

   ```
   type C:\greengrass\v2\\logs\\com.example.HelloWorld.log
   ```

   PowerShell

   ```
   gc C:\greengrass\v2\\logs\\com.example.HelloWorld.log -Tail 10 -Wait
   ```

   Dovresti vedere messaggi simili all'esempio seguente.

   ```
   Hello, universe! Greetings from your first Greengrass component.
   ```

   > **Note**
   >
   > Se i messaggi di registro non vengono modificati, la distribuzione non è riuscita o non ha raggiunto il dispositivo principale. Ciò può verificarsi se il dispositivo principale non è connesso a Internet o non dispone delle autorizzazioni per recuperare artefatti dal bucket S3. Esegui il seguente comando sul tuo dispositivo principale per visualizzare il file di registro del software Core. AWS IoT Greengrass Questo file include i log del servizio di distribuzione del dispositivo principale Greengrass.

Linux or Unix

```
sudo tail -f /greengrass/v2/logs/greengrass.log
```

Windows Command Prompt (CMD)

```
type C:\greengrass\v2\\logs\\greengrass.log
```

Il `type` comando scrive il contenuto del file nel terminale. Esegui questo comando più volte per osservare le modifiche nel file.

PowerShell

```
gc C:\greengrass\v2\\logs\\greengrass.log -Tail 10 -Wait
```

Per ulteriori informazioni, consulta Risoluzione dei problemi AWS IoT Greengrass V2.

Fasi successive

Hai completato questo tutorial. Il softwareAWS IoT Greengrass Core e il componente Hello World vengono eseguiti sul dispositivo. Inoltre, il componente Hello World è disponibile nel servizioAWS IoT Greengrass cloud per essere distribuito su altri dispositivi. Per ulteriori informazioni sui seguenti argomenti, consulta i seguenti argomenti:

- Crea AWS IoT Greengrass componenti
- Pubblica componenti da distribuire sui tuoi dispositivi principali
- Implementazione AWS IoT Greengrass dei componenti sui dispositivi

Configurazione dei dispositivi AWS IoT Greengrass principali

Completa le attività in questa sezione per installare, configurare ed eseguire il software AWS IoT Greengrass Core.

> **Note**
>
> Questa sezione descrive l'installazione e la configurazione avanzate del software AWS IoT Greengrass Core. Se sei un utente alle prime armi diAWS IoT Greengrass V2, ti consigliamo di completare innanzitutto il tutorial introduttivo per configurare un dispositivo principale ed esplorare le funzionalità diAWS IoT Greengrass.

Piattaforme supportate e requisiti

Prima di iniziare, assicurati di soddisfare i seguenti requisiti per installare ed eseguire il software AWS IoT Greengrass Core.

> **Tip**
>
> Puoi cercare i dispositivi idonei AWS IoT Greengrass V2 nel AWSPartner Device Catalog.

Argomenti

Piattaforme supportate

AWS IoT Greengrasssupporta ufficialmente i dispositivi che eseguono le seguenti piattaforme. I dispositivi con piattaforme non incluse in questo elenco potrebbero funzionare, ma AWS IoT Greengrass i test vengono eseguiti solo su queste piattaforme specificate.

Linux

Architetture:

- Armv7l
- Armv8 (AArch64)
- x86_64

Windows

Architetture:

- x86_64

Versioni:

- Windows 10
- Windows 11
- Windows Server 2019
- Windows Server 2022

> **Note**
>
> Alcune AWS IoT Greengrass funzionalità non sono attualmente supportate nei dispositivi Windows. Per ulteriori informazioni, consultare Compatibilità delle funzionalità Greengrass per sistema operativo e Considerazioni sulle funzionalità per i dispositivi Windows.

Le piattaforme Linux possono essere eseguite anche AWS IoT Greengrass V2 in un contenitore Docker. Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core in un contenitore Docker.

Per creare un sistema operativo personalizzato basato su Linux, puoi usare la BitBake ricetta del progetto. AWS IoT Greengrass V2`meta-aws` Il `meta-aws` progetto fornisce ricette che è possibile utilizzare per sviluppare funzionalità software AWS all'avanguardia in sistemi Linux integrati costruiti con OpenEmbeddedi framework di compilazione di Yocto Project. Il progetto Yocto è un progetto di collaborazione open source che consente di creare sistemi personalizzati basati su Linux per

applicazioni integrate indipendentemente dall'architettura hardware. La BitBake ricetta per AWS IoT Greengrass V2 installare, configurare ed eseguire automaticamente il software Core sul dispositivo. AWS IoT Greengrass

Requisiti per il dispositivo

I dispositivi devono soddisfare i seguenti requisiti per installare ed eseguire il software AWS IoT Greengrass Core v2.x.

> **Note**
>
> Puoi utilizzare AWS IoT Device Tester for per AWS IoT Greengrass verificare che il tuo dispositivo sia in grado di eseguire il software AWS IoT Greengrass Core e comunicare con. Cloud AWS Per ulteriori informazioni, consulta Utilizzo AWS IoT Device Tester per AWS IoT Greengrass V2.

Linux

- L'uso di un dispositivo Regione AWSche supportaAWS IoT Greengrass V2. Per l'elenco delle regioni supportate, consulta Endpoint e quote AWS IoT Greengrass V2 in Riferimenti generali di AWS.
- Almeno 256 MB di spazio su disco disponibile per il software AWS IoT Greengrass Core. Questo requisito non include i componenti distribuiti sul dispositivo principale.
- Almeno 96 MB di RAM assegnati al software AWS IoT Greengrass Core. Questo requisito non include i componenti che funzionano sul dispositivo principale. Per ulteriori informazioni, consulta Controlla l'allocazione della memoria con le opzioni JVM.
- Java Runtime Environment (JRE) versione 8 o successiva. Java deve essere disponibile nella variabile di ambiente PATH del dispositivo. Per utilizzare Java per sviluppare componenti personalizzati, è necessario installare un Java Development Kit (JDK). Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva.
- GNU C Library (glibc) versione 2.25 o successiva.
- È necessario eseguire il software AWS IoT Greengrass Core come utente root. Usa`sudo`, ad esempio.
- L'utente root che esegue il software AWS IoT Greengrass Core, ad esempio`root`, deve avere il permesso di funzionare `sudo` con qualsiasi utente e gruppo. Il `/etc/sudoers`

file deve concedere a questo utente l'autorizzazione a funzionare `sudo` come altri gruppi. L'autorizzazione per l'utente `/etc/sudoers` dovrebbe essere simile all'esempio seguente.

```
root    ALL=(ALL:ALL) ALL
```

- Il dispositivo principale deve essere in grado di eseguire richieste in uscita verso un insieme di endpoint e porte. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.
- La `/tmp` directory deve essere montata con `exec` le autorizzazioni.
- Tutti i seguenti comandi di shell:
  - `ps -ax -o pid,ppid`
  - `sudo`
  - `sh`
  - `kill`
  - `cp`
  - `chmod`
  - `rm`
  - `ln`
  - `echo`
  - `exit`
  - `id`
  - `uname`
  - `grep`
- Il dispositivo potrebbe richiedere anche i seguenti comandi shell opzionali:
  - (Facoltativo) `systemctl`. Questo comando viene utilizzato per configurare il software AWS IoT Greengrass Core come servizio di sistema.
  - (`useradd`Facoltativo`groupadd`) e`usermod`. Questi comandi vengono utilizzati per configurare l'`ggc_user`utente e il gruppo `ggc_group` di sistema.
  - (Facoltativo) `mkfifo`. Questo comando viene utilizzato per eseguire funzioni Lambda come componenti.
- Per configurare i limiti delle risorse di sistema per i processi dei componenti, il dispositivo deve eseguire la versione del kernel Linux 2.6.24 o successiva.

- Per eseguire le funzioni Lambda, il dispositivo deve soddisfare requisiti aggiuntivi. Per ulteriori informazioni, consulta Requisiti della funzione Lambda.

Windows

- L'uso di un dispositivo Regione AWSche supportiAWS IoT Greengrass V2. Per l'elenco delle regioni supportate, consulta Endpoint e quote AWS IoT Greengrass V2 in Riferimenti generali di AWS.
- Almeno 256 MB di spazio su disco disponibile per il software AWS IoT Greengrass Core. Questo requisito non include i componenti distribuiti sul dispositivo principale.
- Almeno 160 MB di RAM assegnati al software AWS IoT Greengrass Core. Questo requisito non include i componenti che funzionano sul dispositivo principale. Per ulteriori informazioni, consulta Controlla l'allocazione della memoria con le opzioni JVM.
- Java Runtime Environment (JRE) versione 8 o successiva. Java deve essere disponibile nella variabile di sistema PATH del dispositivo. Per utilizzare Java per sviluppare componenti personalizzati, è necessario installare un Java Development Kit (JDK). Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva.

  > **Note**
  >
  > Per utilizzare la versione 2.5.0 del nucleo Greengrass, è necessario utilizzare una versione a 64 bit di Java Runtime Environment (JRE). La versione 2.5.1 di Greengrass nucleus supporta JRE a 32 e 64 bit.

- L'utente che installa il software AWS IoT Greengrass Core deve essere un amministratore.
- È necessario installare il software AWS IoT Greengrass Core come servizio di sistema. Specificare `--setup-system-service true` quando si installa il software.
- Ogni utente che esegue i processi dei componenti deve esistere nell' LocalSystem account e il nome e la password dell'utente devono trovarsi nell'istanza di Credential Manager dell' LocalSystem account. Puoi configurare questo utente seguendo le istruzioni per installare il software AWS IoT Greengrass Core.
- Il dispositivo principale deve essere in grado di eseguire richieste in uscita verso un insieme di endpoint e porte. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Requisiti della funzione Lambda

Il dispositivo deve soddisfare i seguenti requisiti per eseguire le funzioni Lambda:

- Un sistema operativo basato su Linux.
- Il dispositivo deve avere il comando shell. `mkfifo`
- Il dispositivo deve eseguire le librerie di linguaggi di programmazione richieste da una funzione Lambda. È necessario installare le librerie richieste sul dispositivo e aggiungerle alla variabile di `PATH` ambiente. Greengrass supporta tutte le versioni supportate da Lambda dei runtime Python, Node.js e Java. Greengrass non applica alcuna restrizione aggiuntiva alle versioni di runtime Lambda obsolete. Per ulteriori informazioni sul AWS IoT Greengrass supporto per i runtime Lambda, consulta. Esegui AWS Lambda funzioni
- Per eseguire funzioni Lambda containerizzate, il dispositivo deve soddisfare i seguenti requisiti:
  - Kernel Linux 4.4 o versioni successive.
  - Il kernel deve supportare cgroups v1 ed è necessario abilitare e montare i seguenti cgroup:
    - Il cgroup di memoria AWS IoT Greengrass per impostare il limite di memoria per le funzioni Lambda containerizzate.
    - Il gruppo di dispositivi per le funzioni Lambda containerizzate per accedere ai dispositivi o ai volumi di sistema.

    Il software AWS IoT Greengrass Core non supporta cgroups v2.

    Per soddisfare questo requisito, avvia il dispositivo con i seguenti parametri del kernel Linux.

    ```
    cgroup_enable=memory cgroup_memory=1 systemd.unified_cgroup_hierarchy=0
    ```

    > **Tip**
    >
    > Su un Raspberry Pi, modifica il `/boot/cmdline.txt` file per impostare i parametri del kernel del dispositivo.

  - È necessario abilitare le seguenti configurazioni del kernel Linux sul dispositivo:
    - Spazio dei nomi:
      - `CONFIG_IPC_NS`
      - `CONFIG_UTS_NS`
      - `CONFIG_USER_NS`

    - `CONFIG_PID_NS`
- Cgroups:
    - `CONFIG_CGROUP_DEVICE`
    - `CONFIG_CGROUPS`
    - `CONFIG_MEMCG`
- Altri:
    - `CONFIG_POSIX_MQUEUE`
    - `CONFIG_OVERLAY_FS`
    - `CONFIG_HAVE_ARCH_SECCOMP_FILTER`
    - `CONFIG_SECCOMP_FILTER`
    - `CONFIG_KEYS`
    - `CONFIG_SECCOMP`
    - `CONFIG_SHMEM`

> **Tip**
>
> Consulta la documentazione della tua distribuzione Linux per scoprire come verificare e impostare i parametri del kernel Linux. Puoi anche usare AWS IoT Device Tester for per AWS IoT Greengrass verificare che il tuo dispositivo soddisfi questi requisiti. Per ulteriori informazioni, consulta Utilizzo AWS IoT Device Tester per AWS IoT Greengrass V2.

Considerazioni sulle funzionalità per i dispositivi Windows

Alcune AWS IoT Greengrass funzionalità non sono attualmente supportate nei dispositivi Windows. Esamina le differenze tra le funzionalità per confermare se un dispositivo Windows soddisfa i tuoi requisiti. Per ulteriori informazioni, consulta Compatibilità delle funzionalità Greengrass per sistema operativo.

Configura un Account AWS

Se non disponi di un Account AWS, completa la procedura seguente per crearne uno.

Per registrarsi a un Account AWS

1. Apri la pagina all'indirizzo https://portal.aws.amazon.com/billing/signup.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

   Durante la registrazione di un Account AWS, viene creato un Utente root dell'account AWS. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWSnell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente amministrativo e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

Per creare un utente amministratore, scegli una delle seguenti opzioni.

Scelta di un modo per gestire il tuo amministratore	Per	Come	Puoi anche
In IAM Identity Center (Consigliato)	Usa credenziali a breve termine per accedere a AWS. Ciò è in linea con le best practice per la sicurezza. Per informazioni sulle best practice, consulta Best practice per la sicurezza in IAM nella Guida per l'utente di IAM.	Segui le istruzioni riportate in Nozioni di base nella Guida per l'utente di AWS IAM Identity Center.	Configura l'accesso programmatico seguendo quanto riportato in Configurazione della AWS CLI per utilizzare AWS IAM Identity Center nella Guida per l'utente di AWS Command Line Interface.

Scelta di un modo per gestire il tuo amministratore	Per	Come	Puoi anche
In IAM (Non consigliato)	Usa credenziali a lungo termine per accedere a AWS.	Segui le istruzioni in Creazione del primo utente e gruppo di utenti IAM di amministrazione nella Guida per l'utente di IAM.	Configura l'accesso programmatico seguendo quanto riportato in Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente di IAM.

Installare il software AWS IoT Greengrass Core.

AWS IoT Greengrasssi estende AWS ai dispositivi periferici in modo che possano agire sui dati generati, mentre li utilizzano Cloud AWS per la gestione, l'analisi e lo storage durevole. Installa il software AWS IoT Greengrass Core sui dispositivi periferici da integrare con AWS IoT Greengrass eCloud AWS.

> **Important**
>
> Prima di scaricare e installare il software AWS IoT Greengrass Core, verifica che il dispositivo principale soddisfi i requisiti per installare ed eseguire il software AWS IoT Greengrass Core v2.0.

Il software AWS IoT Greengrass Core include un programma di installazione che configura il dispositivo come dispositivo principale Greengrass. Quando esegui il programma di installazione, puoi configurare le opzioni, come la cartella principale e da utilizzare. Regione AWS Puoi scegliere di fare in modo che il programma di installazione crei le risorse IAM necessarie AWS IoT e necessarie per te. Puoi anche scegliere di implementare strumenti di sviluppo locali per configurare un dispositivo da utilizzare per lo sviluppo di componenti personalizzati.

Il software AWS IoT Greengrass Core richiede le seguenti risorse AWS IoT e quelle di IAM per connettersi Cloud AWS e funzionare:

- Un oggetto AWS IoT Quando si registra un dispositivo come AWS IoT oggetto, quel dispositivo può utilizzare un certificato digitale con AWS cui autenticarsi. Questo certificato consente al dispositivo di comunicare con AWS IoT eAWS IoT Greengrass. Per ulteriori informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass.
- (Facoltativo) Qualsiasi gruppo di AWS IoT cose. Utilizzi i gruppi di cose per gestire flotte di dispositivi core Greengrass. Quando distribuisci componenti software sui tuoi dispositivi, puoi scegliere di distribuirli su singoli dispositivi o su gruppi di dispositivi. È possibile aggiungere un dispositivo a un gruppo di oggetti per distribuire i componenti software di quel gruppo di oggetti sul dispositivo. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.
- Un ruolo IAM. I dispositivi core Greengrass utilizzano il provider di AWS IoT Core credenziali per autorizzare le chiamate ai AWS servizi con un ruolo IAM. Questo ruolo consente al dispositivo di interagireAWS IoT, inviare log ad Amazon Logs e scaricare elementi di componenti personalizzati da Amazon CloudWatch Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3). Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.
- Un alias di AWS IoT ruolo. I dispositivi core Greengrass utilizzano l'alias del ruolo per identificare il ruolo IAM da utilizzare. L'alias del ruolo consente di modificare il ruolo IAM mantenendo invariata la configurazione del dispositivo. Per ulteriori informazioni, consulta Autorizzazione delle chiamate dirette ai AWS servizi nella Guida per gli AWS IoT Coresviluppatori.

Scegli una delle seguenti opzioni per installare il software AWS IoT Greengrass Core sul tuo dispositivo principale.

- Installazione rapida

  Scegli questa opzione per configurare un dispositivo Greengrass core nel minor numero di passaggi possibile. Il programma di installazione crea per te le risorse IAM necessarieAWS IoT. Questa opzione richiede che tu fornisca AWS le credenziali all'installatore per creare risorse nel tuo. Account AWS

  Non è possibile utilizzare questa opzione per l'installazione dietro un firewall o un proxy di rete. Se i tuoi dispositivi sono protetti da un firewall o da un proxy di rete, prendi in considerazione l'installazione manuale.

Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning automatico delle risorse.

- Installazione manuale

  Scegli questa opzione per creare le AWS risorse richieste manualmente o per installarle dietro un firewall o un proxy di rete. Utilizzando un'installazione manuale, non è necessario concedere all'installatore l'autorizzazione per creare risorse nel proprio computerAccount AWS, in quanto si creano le risorse necessarie AWS IoT e quelle IAM. Puoi anche configurare il dispositivo in modo che si connetta alla porta 443 o tramite un proxy di rete. Puoi anche configurare il software AWS IoT Greengrass Core per utilizzare una chiave privata e un certificato archiviati in un modulo di sicurezza hardware (HSM), Trusted Platform Module (TPM) o un altro elemento crittografico.

  Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning manuale delle risorse.

- Installazione con provisioning della flotta AWS IoT

  Scegli questa opzione per creare le AWS risorse richieste da un modello di approvvigionamento AWS IoT della flotta. Puoi scegliere questa opzione per creare dispositivi simili in una flotta o se produci dispositivi che i tuoi clienti attiveranno successivamente, come veicoli o dispositivi per la casa intelligente. I dispositivi utilizzano certificati di attestazione per autenticare e fornire AWS risorse, incluso un certificato client X.509 che il dispositivo utilizza per connettersi al sistema Cloud AWS per il normale funzionamento. È possibile incorporare o aggiornare i certificati di attestazione nell'hardware del dispositivo durante la produzione e utilizzare lo stesso certificato di attestazione e la stessa chiave per effettuare il provisioning di più dispositivi. È inoltre possibile configurare i dispositivi in modo che si connettano alla porta 443 o tramite un proxy di rete.

  Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con il provisioning AWS IoT della flotta.

- Installazione con provisioning personalizzato

  Scegliete questa opzione per sviluppare un'applicazione Java personalizzata che fornisca le AWS risorse necessarie. È possibile scegliere questa opzione se si creano certificati client X.509 personalizzati o se si desidera un maggiore controllo sul processo di provisioning. AWS IoT Greengrassfornisce un'interfaccia che è possibile implementare per lo scambio di informazioni tra l'applicazione di provisioning personalizzata e il programma di installazione del AWS IoT Greengrass software Core.

Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning personalizzato delle risorse.

AWS IoT Greengrass fornisce anche ambienti containerizzati che eseguono il software AWS IoT Greengrass Core. È possibile utilizzare un Dockerfile per eseguirlo AWS IoT Greengrass in un contenitore Docker.

Argomenti

Installa il software AWS IoT Greengrass Core con provisioning automatico delle risorse

Il software AWS IoT Greengrass Core include un programma di installazione che configura il dispositivo come dispositivo principale Greengrass. Per configurare rapidamente un dispositivo, l'installatore può fornire il thing, il AWS IoT AWS IoT thing group, il ruolo IAM e l'alias di AWS IoT ruolo necessari per il funzionamento del dispositivo principale. Il programma di installazione può anche distribuire gli strumenti di sviluppo locali sul dispositivo principale, in modo da poter utilizzare il dispositivo per sviluppare e testare componenti software personalizzati. Il programma di installazione richiede AWS credenziali per fornire queste risorse e creare la distribuzione.

Se non è possibile fornire AWS le credenziali al dispositivo, è possibile fornire le AWS risorse necessarie al funzionamento del dispositivo principale. Puoi anche distribuire gli strumenti di sviluppo su un dispositivo principale da utilizzare come dispositivo di sviluppo. Ciò consente di fornire meno autorizzazioni al dispositivo quando si esegue il programma di installazione. Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning manuale delle risorse.

> **Important**
>
> Prima di scaricare il software AWS IoT Greengrass Core, verifica che il dispositivo principale soddisfi i requisiti per installare ed eseguire il software AWS IoT Greengrass Core v2.0.

Argomenti

Configura l'ambiente del dispositivo

Segui i passaggi di questa sezione per configurare un dispositivo Linux o Windows da utilizzare come dispositivo AWS IoT Greengrass principale.

Configura un dispositivo Linux

Per configurare un dispositivo Linux per AWS IoT Greengrass V2

1. Installa il runtime Java, necessario per l'esecuzione del software AWS IoT Greengrass Core. Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva. I seguenti comandi mostrano come installare OpenJDK sul tuo dispositivo.

   - Per le distribuzioni basate su Debian o basate su Ubuntuu:

     ```
     sudo apt install default-jdk
     ```

   - Per le distribuzioni basate su Red Hat:

     ```
     sudo yum install java-11-openjdk-devel
     ```

   - Per Amazon Linux 2:

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - Per Amazon Linux 2023:

     ```
     sudo dnf install java-11-amazon-corretto -y
     ```

   Al termine dell'installazione, esegui il comando seguente per verificare che Java sia in esecuzione sul tuo dispositivo Linux.

```
java -version
```

Il comando stampa la versione di Java in esecuzione sul dispositivo. Ad esempio, su una distribuzione basata su Debian, l'output potrebbe essere simile all'esempio seguente.

```
openjdk version "11.0.9.1" 2020-11-04
OpenJDK Runtime Environment (build 11.0.9.1+1-post-Debian-1deb10u2)
OpenJDK 64-Bit Server VM (build 11.0.9.1+1-post-Debian-1deb10u2, mixed mode)
```

2. (Facoltativo) Crea l'utente e il gruppo di sistema predefiniti che eseguono i componenti sul dispositivo. Puoi anche scegliere di lasciare che il programma di installazione del software AWS IoT Greengrass Core crei questo utente e gruppo durante l'installazione con l'argomento `--component-default-user` installer. Per ulteriori informazioni, consulta Argomenti dell'installatore.

```
sudo useradd --system --create-home ggc_user
sudo groupadd --system ggc_group
```

3. Verificate che l'utente che esegue il software AWS IoT Greengrass Core (in genere`root`) sia autorizzato a funzionare `sudo` con qualsiasi utente e gruppo.

   a. Eseguite il comando seguente per aprire il `/etc/sudoers` file.

   ```
   sudo visudo
   ```

   b. Verificate che l'autorizzazione per l'utente sia simile all'esempio seguente.

   ```
   root    ALL=(ALL:ALL) ALL
   ```

4. (Facoltativo) Per eseguire funzioni Lambda containerizzate, è necessario abilitare cgroups v1 e abilitare e montare i cgroup di memoria e dispositivi. Se non intendi eseguire funzioni Lambda containerizzate, puoi saltare questo passaggio.

   Per abilitare queste opzioni di cgroups, avvia il dispositivo con i seguenti parametri del kernel Linux.

   ```
   cgroup_enable=memory cgroup_memory=1 systemd.unified_cgroup_hierarchy=0
   ```

Per informazioni sulla visualizzazione e l'impostazione dei parametri del kernel per il tuo dispositivo, consulta la documentazione del tuo sistema operativo e del boot loader. Segui le istruzioni per impostare in modo permanente i parametri del kernel.

5. Installa tutte le altre dipendenze richieste sul tuo dispositivo come indicato dall'elenco dei requisiti in. Requisiti per il dispositivo

Configura un dispositivo Windows

> **Note**
>
> Questa funzionalità è disponibile per la versione 2.5.0 e successive del componente Greengrass nucleus.

Per configurare un dispositivo Windows per AWS IoT Greengrass V2

1. Installa il runtime Java, necessario per l'esecuzione del software AWS IoT Greengrass Core. Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva.
2. Controlla se Java è disponibile nella variabile di sistema PATH e aggiungilo in caso contrario. L' LocalSystem account esegue il software AWS IoT Greengrass Core, quindi è necessario aggiungere Java alla variabile di sistema PATH anziché alla variabile utente PATH per l'utente. Esegui questa operazione:

   a. Premi il tasto Windows per aprire il menu di avvio.

   b. Digita **`environment variables`** per cercare le opzioni di sistema dal menu di avvio.

   c. Nei risultati della ricerca del menu di avvio, scegli Modifica le variabili di ambiente di sistema per aprire la finestra delle proprietà del sistema.

   d. Scegli le variabili di ambiente... per aprire la finestra Variabili d'ambiente.

   e. In Variabili di sistema, seleziona Percorso, quindi scegli Modifica. Nella finestra Modifica variabile di ambiente, puoi visualizzare ogni percorso su una riga separata.

   f. Controlla se è presente il percorso della `bin` cartella di installazione di Java. Il percorso potrebbe essere simile all'esempio seguente.

   ```
   C:\\Program Files\\Amazon Corretto\\jdk11.0.13_8\\bin
   ```

g. Se la `bin` cartella di installazione Java non è presente in Path, scegliete Nuovo per aggiungerla, quindi scegliete OK.

3. Aprite il prompt dei comandi di Windows (`cmd.exe`) come amministratore.
4. Crea l'utente predefinito nell' LocalSystem account sul dispositivo Windows. Sostituisci *la password* con una password sicura.

```
net user /add ggc_user password
```

> **Tip**
>
> A seconda della configurazione di Windows, la password dell'utente potrebbe essere impostata per scadere in date future. Per garantire che le tue applicazioni Greengrass continuino a funzionare, tieni traccia della scadenza della password e aggiornala prima che scada. Puoi anche impostare la password dell'utente in modo che non scada mai.
>
> - Per verificare la scadenza di un utente e della relativa password, esegui il comando seguente.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Per impostare la password di un utente in modo che non scada mai, esegui il comando seguente.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```
>
> - Se utilizzi Windows 10 o versioni successive in cui il wmiccomando è obsoleto, esegui il comando seguente. PowerShell
>
>   ```
>   Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
>    'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
>   ```

5. Scarica e installa l'PsExecutilità di Microsoft sul dispositivo.
6. Utilizzate l' PsExec utilità per memorizzare il nome utente e la password per l'utente predefinito nell'istanza di Credential Manager per l' LocalSystem account. Sostituisci la *password* con la password dell'utente impostata in precedenza.

```
psexec -s cmd /c cmdkey /generic:ggc_user /user:ggc_user /pass:password
```

Se si PsExec License Agreementapre, scegli Acceptdi accettare la licenza ed esegui il comando.

> **Note**
>
> Sui dispositivi Windows, l' LocalSystem account esegue il Greengrass nucleus ed è necessario utilizzare l' PsExec utilità per memorizzare le informazioni utente predefinite nell'account. LocalSystem L'utilizzo dell'applicazione Credential Manager archivia queste informazioni nell'account Windows dell'utente attualmente connesso, anziché nell'account. LocalSystem

Fornisci AWS le credenziali al dispositivo

Fornisci AWS le tue credenziali al dispositivo in modo che l'installatore possa fornire le risorse necessarie. AWS Per ulteriori informazioni sulle autorizzazioni richieste, consulta Policy IAM minima per l'installatore per il provisioning delle risorse.

Per fornire AWS le credenziali al dispositivo

- Fornisci AWS le tue credenziali al dispositivo in modo che l'installatore possa fornire le risorse AWS IoT e IAM per il tuo dispositivo principale. Per aumentare la sicurezza, ti consigliamo di ottenere credenziali temporanee per un ruolo IAM che consenta solo le autorizzazioni minime necessarie per il provisioning. Per ulteriori informazioni, consulta Policy IAM minima per l'installatore per il provisioning delle risorse.

  > **Note**
  >
  > Il programma di installazione non salva né archivia le tue credenziali.

  Sul dispositivo, esegui una delle seguenti operazioni per recuperare le credenziali e renderle disponibili al programma di installazione del AWS IoT Greengrass software Core:

  - (Consigliato) Utilizza credenziali temporanee da AWS IAM Identity Center

    a. Fornisci l'ID della chiave di accesso, la chiave di accesso segreta e il token di sessione dall'IAM Identity Center. Per ulteriori informazioni, consulta Aggiornamento manuale delle credenziali in Acquisizione e aggiornamento delle credenziali temporanee nella guida per l'utente di IAM Identity Center.

b. Esegui i seguenti comandi per fornire le credenziali al software Core. AWS IoT Greengrass

Linux or Unix

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_SESSION_TOKEN=AQoDYXdzEJr1K...o5OytwEXAMPLE=
```

Windows Command Prompt (CMD)

```
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
set AWS_SESSION_TOKEN=AQoDYXdzEJr1K...o5OytwEXAMPLE=
```

PowerShell

```
$env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
$env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
$env:AWS_SESSION_TOKEN="AQoDYXdzEJr1K...o5OytwEXAMPLE="
```

- Utilizza credenziali di sicurezza temporanee da un ruolo IAM:

a. Fornisci l'ID della chiave di accesso, la chiave di accesso segreta e il token di sessione da un ruolo IAM che assumi. Per ulteriori informazioni su come recuperare queste credenziali, consulta la sezione Richiesta di credenziali di sicurezza temporanee nella Guida per l'utente IAM.

b. Esegui i seguenti comandi per fornire le credenziali al software Core. AWS IoT Greengrass

Linux or Unix

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
export AWS_SESSION_TOKEN=AQoDYXdzEJr1K...o5OytwEXAMPLE=
```

Windows Command Prompt (CMD)

```
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

```
set AWS_SESSION_TOKEN=AQoDYXdzEJr1K...o5OytwEXAMPLE=
```

PowerShell

```
$env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
$env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
$env:AWS_SESSION_TOKEN="AQoDYXdzEJr1K...o5OytwEXAMPLE="
```

- Utilizza le credenziali a lungo termine di un utente IAM:

  a. Fornisci l'ID della chiave di accesso e la chiave di accesso segreta per il tuo utente IAM. Puoi creare un utente IAM per il provisioning da eliminare successivamente. Per la policy IAM da fornire all'utente, consultaPolicy IAM minima per l'installatore per il provisioning delle risorse. Per ulteriori informazioni su come recuperare le credenziali a lungo termine, consulta Managing access keys for IAM users nella IAM User Guide.

  b. Esegui i seguenti comandi per fornire le credenziali al AWS IoT Greengrass software Core.

  Linux or Unix

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

  Windows Command Prompt (CMD)

```
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

  PowerShell

```
$env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
$env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
```

  c. (Facoltativo) Se hai creato un utente IAM per il provisioning del tuo dispositivo Greengrass, elimina l'utente.

  d. (Facoltativo) Se hai utilizzato l'ID della chiave di accesso e la chiave di accesso segreta di un utente IAM esistente, aggiorna le chiavi dell'utente in modo che non siano più valide. Per ulteriori informazioni, consulta Aggiornamento delle chiavi di accesso nella guida AWS Identity and Access Management per l'utente.

Scaricate il software AWS IoT Greengrass Core

È possibile scaricare la versione più recente del software AWS IoT Greengrass Core dal seguente indirizzo:

- https://d2s8p88vqu9w66.cloudfront.net/releases/ greengrass-nucleus-latest .zip

> **Note**
>
> È possibile scaricare una versione specifica del software AWS IoT Greengrass Core dal seguente percorso. Sostituisci la *`versione`* con la versione da scaricare.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Per scaricare il software AWS IoT Greengrass Core

1. Sul dispositivo principale, scaricate il software AWS IoT Greengrass Core in un file denominato`greengrass-nucleus-latest.zip`.

   Linux or Unix

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
   latest.zip > greengrass-nucleus-latest.zip
   ```

   Windows Command Prompt (CMD)

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
   latest.zip > greengrass-nucleus-latest.zip
   ```

   PowerShell

   ```
   iwr -Uri https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
   latest.zip -OutFile greengrass-nucleus-latest.zip
   ```

   Scaricando questo software accetti l'Accordo di licenza del software Greengrass Core.

2. (Facoltativo) Per verificare la firma del software Greengrass nucleus

> **Note**
>
> Questa funzionalità è disponibile con Greengrass nucleus versione 2.9.5 e successive.

a. Usa il seguente comando per verificare la firma del tuo artefatto Greengrass nucleus:

   Linux or Unix

   ```
   jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
   ```

   Windows Command Prompt (CMD)

   Il nome del file potrebbe avere un aspetto diverso a seconda della versione di JDK installata. *jdk17.0.6_10*Sostituiscilo con la versione JDK che hai installato.

   ```
   "C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
   verify -certs -verbose greengrass-nucleus-latest.zip
   ```

   PowerShell

   Il nome del file potrebbe avere un aspetto diverso a seconda della versione di JDK installata. *jdk17.0.6_10*Sostituiscilo con la versione JDK che hai installato.

   ```
   'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
   verify -certs -verbose greengrass-nucleus-latest.zip
   ```

b. L'`jarsigner`invocazione produce un output che indica i risultati della verifica.

   i. Se il file zip Greengrass nucleus è firmato, l'output contiene la seguente dichiarazione:

   ```
   jar verified.
   ```

   ii. Se il file zip Greengrass nucleus non è firmato, l'output contiene la seguente dichiarazione:

   ```
   jar is unsigned.
   ```

c. Se hai fornito l'`-certs`opzione Jarsigner insieme alle `-verbose` opzioni `-verify` e, l'output include anche informazioni dettagliate sul certificato del firmatario.

3. Decomprimi il software AWS IoT Greengrass Core in una cartella sul tuo dispositivo. Sostituiscilo *`GreengrassInstaller`*con la cartella che desideri utilizzare.

   Linux or Unix

   ```
   unzip greengrass-nucleus-latest.zip -d GreengrassInstaller && rm greengrass-
   nucleus-latest.zip
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir GreengrassInstaller && tar -xf greengrass-nucleus-latest.zip -
   C GreengrassInstaller && del greengrass-nucleus-latest.zip
   ```

   PowerShell

   ```
   Expand-Archive -Path greengrass-nucleus-latest.zip -DestinationPath .\
   \GreengrassInstaller
   rm greengrass-nucleus-latest.zip
   ```

4. (Facoltativo) Eseguite il comando seguente per visualizzare la versione del software AWS IoT Greengrass Core.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

> **Important**
>
> Se installi una versione del nucleo Greengrass precedente alla v2.4.0, non rimuovere questa cartella dopo aver installato il software Core. AWS IoT Greengrass Il software AWS IoT Greengrass Core utilizza i file in questa cartella per l'esecuzione.
> Se hai scaricato la versione più recente del software, installi la versione 2.4.0 o successiva e puoi rimuovere questa cartella dopo aver installato il software AWS IoT Greengrass Core.

Installare il software AWS IoT Greengrass Core.

Eseguite il programma di installazione con argomenti che specificano di effettuare le seguenti operazioni:

- AWSCreate le risorse necessarie al funzionamento del dispositivo principale.
- Specificare di utilizzare l'utente `ggc_user` del sistema per eseguire i componenti software sul dispositivo principale. Sui dispositivi Linux, questo comando specifica anche di utilizzare il gruppo di `ggc_group` sistema e il programma di installazione crea automaticamente l'utente e il gruppo di sistema.
- Configura il software AWS IoT Greengrass Core come servizio di sistema che viene eseguito all'avvio. Sui dispositivi Linux, ciò richiede il sistema di inizializzazione Systemd.

> **Important**
>
> Sui dispositivi Windows core, è necessario configurare il software AWS IoT Greengrass Core come servizio di sistema.

Per configurare un dispositivo di sviluppo con strumenti di sviluppo locali, specificate l'`--deploy-dev-tools true`argomento. L'implementazione degli strumenti di sviluppo locale può richiedere fino a un minuto dopo il completamento dell'installazione.

Per ulteriori informazioni sugli argomenti che è possibile specificare, vedere. Argomenti dell'installatore

> **Note**
>
> Se utilizzi AWS IoT Greengrass un dispositivo con memoria limitata, puoi controllare la quantità di memoria utilizzata dal software AWS IoT Greengrass Core. Per controllare l'allocazione della memoria, è possibile impostare le opzioni relative alla dimensione dell'heap JVM nel parametro di `jvmOptions` configurazione del componente nucleus. Per ulteriori informazioni, consulta Controlla l'allocazione della memoria con le opzioni JVM.

Installare il software AWS IoT Greengrass Core.

1. Esegui il programma di installazione Core. AWS IoT Greengrass Sostituisci i valori degli argomenti nel tuo comando come segue.

   a. *`/greengrass/v2`*o *`C:\greengrass\v2`*: il percorso della cartella principale da utilizzare per installare il software AWS IoT Greengrass Core.

   b. *`GreengrassInstaller`*. Il percorso della cartella in cui è stato decompresso il programma di installazione del software AWS IoT Greengrass Core.

   c. *`regione.`* Il Regione AWS luogo in cui trovare o creare risorse.

   d. *`MyGreengrassCore`*. Il nome del AWS IoT dispositivo principale Greengrass. Se l'oggetto non esiste, l'installatore lo crea. Il programma di installazione scarica i certificati per autenticarsi come oggetto. AWS IoT Per ulteriori informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass.

      > **Note**
      >
      > Il nome dell'oggetto non può contenere i due punti (`:`).

   e. *`MyGreengrassCoreGroup`*. Il nome del AWS IoT gruppo di oggetti per il tuo dispositivo principale Greengrass. Se il gruppo di oggetti non esiste, l'installatore lo crea e vi aggiunge l'oggetto. Se il gruppo di oggetti esiste e dispone di una distribuzione attiva, il dispositivo principale scarica ed esegue il software specificato dalla distribuzione.

      > **Note**
      >
      > Il nome del gruppo di cose non può contenere i due punti (`:`).

   f. *`GreenGrass v2 IoT ThingPolicy`*. Il nome della AWS IoT policy che consente ai dispositivi core Greengrass di comunicare con AWS IoT e. AWS IoT Greengrass Se la AWS IoT politica non esiste, il programma di installazione crea una AWS IoT politica permissiva con questo nome. Puoi limitare le autorizzazioni di questa politica in base al tuo caso d'uso. Per ulteriori informazioni, consulta AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali.

   g. *`GreenGrass v2 TokenExchangeRole`*. Il nome del ruolo IAM che consente al dispositivo principale Greengrass di ottenere credenziali temporaneeAWS. Se il ruolo non esiste, l'installatore lo crea e crea e allega una policy denominata.

*GreengrassV2TokenExchangeRole*Access Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

h. *GreengrassCoreTokenExchangeRoleAlias*. L'alias del ruolo IAM che consente al dispositivo principale Greengrass di ottenere credenziali temporanee in un secondo momento. Se l'alias del ruolo non esiste, il programma di installazione lo crea e lo indirizza al ruolo IAM specificato. Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

Linux or Unix

```
sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
  -jar ./GreengrassInstaller/lib/Greengrass.jar \
  --aws-region region \
  --thing-name MyGreengrassCore \
  --thing-group-name MyGreengrassCoreGroup \
  --thing-policy-name GreengrassV2IoTThingPolicy \
  --tes-role-name GreengrassV2TokenExchangeRole \
  --tes-role-alias-name GreengrassCoreTokenExchangeRoleAlias \
  --component-default-user ggc_user:ggc_group \
  --provision true \
  --setup-system-service true
```

Windows Command Prompt (CMD)

```
java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" ^
  -jar ./GreengrassInstaller/lib/Greengrass.jar ^
  --aws-region region ^
  --thing-name MyGreengrassCore ^
  --thing-group-name MyGreengrassCoreGroup ^
  --thing-policy-name GreengrassV2IoTThingPolicy ^
  --tes-role-name GreengrassV2TokenExchangeRole ^
  --tes-role-alias-name GreengrassCoreTokenExchangeRoleAlias ^
  --component-default-user ggc_user ^
  --provision true ^
  --setup-system-service true
```

PowerShell

```
java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" `
  -jar ./GreengrassInstaller/lib/Greengrass.jar `
```

```
  --aws-region region `
  --thing-name MyGreengrassCore `
  --thing-group-name MyGreengrassCoreGroup `
  --thing-policy-name GreengrassV2IoTThingPolicy `
  --tes-role-name GreengrassV2TokenExchangeRole `
  --tes-role-alias-name GreengrassCoreTokenExchangeRoleAlias `
  --component-default-user ggc_user `
  --provision true `
  --setup-system-service true
```

> **Important**
>
> Nei dispositivi Windows core, è necessario specificare `--setup-system-service true` di configurare il software AWS IoT Greengrass Core come servizio di sistema.

Il programma di installazione stampa i seguenti messaggi se riesce:

- Se si specifica`--provision`, il programma di installazione stampa `Successfully configured Nucleus with provisioned resource details` se ha configurato correttamente le risorse.
- Se si specifica`--deploy-dev-tools`, il programma di installazione stampa `Configured Nucleus to deploy aws.greengrass.Cli component` se ha creato la distribuzione con successo.
- Se si specifica`--setup-system-service true`, il programma di installazione stampa `Successfully set up Nucleus as a system service` se ha configurato ed eseguito il software as a service.
- Se non viene specificato`--setup-system-service true`, il programma di installazione stampa `Launched Nucleus successfully` se l'operazione è riuscita ed ha eseguito il software.

2. Salta questo passaggio se hai installato Nucleo Greengrass la versione 2.0.4 o successiva. Se hai scaricato la versione più recente del software, hai installato la versione 2.0.4 o successiva.

   Esegui il comando seguente per impostare le autorizzazioni di file richieste per la cartella AWS IoT Greengrass principale del software Core. Sostituisci */greengrass/v2* con la cartella principale specificata nel comando di installazione e sostituisci */greengrass* con la cartella principale per la tua cartella principale.

```
sudo chmod 755 /greengrass/v2 && sudo chmod 755 /greengrass
```

Se avete installato il software AWS IoT Greengrass Core come servizio di sistema, il programma di installazione esegue il software per voi. In caso contrario, è necessario eseguire il software manualmente. Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core.

> **Note**
>
> Per impostazione predefinita, il ruolo IAM creato dal programma di installazione non consente l'accesso agli artefatti dei componenti nei bucket S3. Per distribuire componenti personalizzati che definiscono gli artefatti in Amazon S3, devi aggiungere autorizzazioni al ruolo per consentire al dispositivo principale di recuperare gli artefatti dei componenti. Per ulteriori informazioni, consulta Consenti l'accesso ai bucket S3 per gli artefatti dei componenti. Se non disponi ancora di un bucket S3 per gli artefatti dei componenti, puoi aggiungere queste autorizzazioni in un secondo momento dopo aver creato un bucket.

Per ulteriori informazioni su come configurare e utilizzare il software e, consulta quanto segue: AWS IoT Greengrass

- Configurare il software AWS IoT Greengrass Core
- Sviluppa AWS IoT Greengrass componenti
- Implementazione AWS IoT Greengrass dei componenti sui dispositivi
- Interfaccia a riga di comando Greengrass

Installa il software AWS IoT Greengrass Core con provisioning manuale delle risorse

Il software AWS IoT Greengrass Core include un programma di installazione che configura il dispositivo come dispositivo principale Greengrass. Per configurare un dispositivo manualmente, puoi creare le risorse IAM necessarie AWS IoT e le risorse IAM da utilizzare per il dispositivo. Se crei queste risorse manualmente, non è necessario fornire AWS le credenziali all'installatore.

Quando installi manualmente il software AWS IoT Greengrass Core, puoi anche configurare il dispositivo per utilizzare un proxy di rete o connetterti alla porta AWS 443. Potrebbe essere

necessario specificare queste opzioni di configurazione se il dispositivo è protetto da un firewall o da un proxy di rete, ad esempio. Per ulteriori informazioni, consulta Connessione alla porta 443 o tramite un proxy di rete.

È inoltre possibile configurare il software AWS IoT Greengrass Core per utilizzare un modulo di sicurezza hardware (HSM) tramite l'interfaccia PKCS #11. Questa funzionalità consente di archiviare in modo sicuro i file di chiavi e certificati privati in modo che non vengano esposti o duplicati nel software. È possibile archiviare chiavi e certificati privati su un modulo hardware come un HSM, un Trusted Platform Module (TPM) o un altro elemento crittografico. Questa funzionalità è disponibile solo sui dispositivi Linux. Per ulteriori informazioni sulla sicurezza dell'hardware e sui requisiti per utilizzarla, vedereIntegrazione della sicurezza hardware.

> **Important**
>
> Prima di scaricare il software AWS IoT Greengrass Core, verifica che il dispositivo principale soddisfi i requisiti per installare ed eseguire il software AWS IoT Greengrass Core v2.0.

Argomenti

Recupera AWS IoT gli endpoint

Ottieni gli AWS IoT endpoint per te e salvali per Account AWS utilizzarli in un secondo momento. Il tuo dispositivo utilizza questi endpoint per connettersi a. AWS IoT Esegui questa operazione:

1. Ottieni l'endpoint di AWS IoT dati per il tuo. Account AWS

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
}
```

2. Ottieni l'endpoint delle AWS IoT credenziali per il tuo. Account AWS

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "endpointAddress": "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
}
```

Crea qualsiasi cosa AWS IoT

AWS IoT le cose rappresentano dispositivi ed entità logiche a cui si connettono AWS IoT. I dispositivi core Greengrass sono AWS IoT cose. Quando registri un dispositivo come AWS IoT oggetto, quel dispositivo può utilizzare un certificato digitale con cui autenticarsi. AWS

In questa sezione, crei AWS IoT qualcosa che rappresenta il tuo dispositivo.

Per creare qualsiasi AWS IoT cosa

1. Crea AWS IoT qualcosa per il tuo dispositivo. Sul tuo computer di sviluppo, esegui il seguente comando.

   - Sostituisci *MyGreengrassCore*con il nome dell'oggetto da usare. Questo nome è anche il nome del dispositivo principale Greengrass.

     > **Note**
     >
     > Il nome dell'oggetto non può contenere i due punti (`:`).

```
aws iot create-thing --thing-name MyGreengrassCore
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "thingName": "MyGreengrassCore",
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "thingId": "8cb4b6cd-268e-495d-b5b9-1713d71dbf42"
}
```

2. (Facoltativo) Aggiungere l' AWS IoT oggetto a un gruppo di oggetti nuovo o esistente. Utilizzi i gruppi di cose per gestire flotte di dispositivi core Greengrass. Quando distribuisci componenti software sui tuoi dispositivi, puoi scegliere come target singoli dispositivi o gruppi di dispositivi. È possibile aggiungere un dispositivo a un gruppo di cose con una distribuzione Greengrass attiva per distribuire i componenti software di quel gruppo di cose sul dispositivo. Esegui questa operazione:

   a. (Facoltativo) Crea un gruppo di AWS IoT cose.

      - Sostituire *MyGreengrassCoreGroup*con il nome del gruppo di oggetti da creare.

      > **Note**
      >
      > Il nome del gruppo di cose non può contenere i due punti (`:`).

      ```
      aws iot create-thing-group --thing-group-name MyGreengrassCoreGroup
      ```

      La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

      ```
      {
        "thingGroupName": "MyGreengrassCoreGroup",
        "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
      MyGreengrassCoreGroup",
        "thingGroupId": "4df721e1-ff9f-4f97-92dd-02db4e3f03aa"
      }
      ```

   b. Aggiungere l' AWS IoT oggetto a un gruppo di oggetti.

- Sostituiscilo *MyGreengrassCore*con il nome del tuo AWS IoT oggetto.
- Sostituisci *MyGreengrassCoreGroup*con il nome del gruppo di oggetti.

```
aws iot add-thing-to-thing-group --thing-name MyGreengrassCore --thing-group-
name MyGreengrassCoreGroup
```

Il comando non produce alcun output se la richiesta ha esito positivo.

Crea il certificato dell'oggetto

Quando si registra un dispositivo come AWS IoT oggetto, quel dispositivo può utilizzare un certificato digitale con AWS cui autenticarsi. Questo certificato consente al dispositivo di comunicare con AWS IoT e AWS IoT Greengrass.

In questa sezione, crei e scarichi certificati a cui il tuo dispositivo può connettersi AWS.

Se desideri configurare il software AWS IoT Greengrass Core per utilizzare un modulo di sicurezza hardware (HSM) per archiviare in modo sicuro la chiave privata e il certificato, segui i passaggi per creare il certificato da una chiave privata in un HSM. Altrimenti, segui i passaggi per creare il certificato e la chiave privata nel servizio. AWS IoT La funzionalità di sicurezza hardware è disponibile solo sui dispositivi Linux. Per ulteriori informazioni sulla sicurezza dell'hardware e sui requisiti per utilizzarla, vedereIntegrazione della sicurezza hardware.

Creare il certificato e la chiave privata nel AWS IoT servizio

Per creare il certificato dell'oggetto

1. Crea una cartella in cui scaricare i certificati relativi all' AWS IoT oggetto.

   ```
   mkdir greengrass-v2-certs
   ```

2. Crea e scarica i certificati relativi all' AWS IoT oggetto.

   ```
   aws iot create-keys-and-certificate --set-as-active --certificate-pem-outfile
    greengrass-v2-certs/device.pem.crt --public-key-outfile greengrass-v2-certs/
   public.pem.key --private-key-outfile greengrass-v2-certs/private.pem.key
   ```

   La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
  "certificateId":
 "aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
  "certificatePem": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
  "keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----\
MIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\
MMEXAMPLEuuN/dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/gHr99VEEXAMPLE5VF13\
59VK7cEXAMPLE67GK+y+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQWE
\\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa\
hJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\
FQIDAQAB\
-----END PUBLIC KEY-----\
",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\
key omitted for security reasons\
-----END RSA PRIVATE KEY-----\
"
  }
}
```

Salva l'Amazon Resource Name (ARN) del certificato da utilizzare per configurare il certificato in un secondo momento.

Crea il certificato da una chiave privata in un HSM

> **Note**
>
> Questa funzionalità è disponibile per la versione 2.5.3 e successive del componente Greengrass nucleus. AWS IoT Greengrass attualmente non supporta questa funzionalità sui dispositivi Windows core.

Per creare il certificato Thing

1. Sul dispositivo principale, inizializza un token PKCS #11 nell'HSM e genera una chiave privata. La chiave privata deve essere una chiave RSA con una dimensione di chiave RSA-2048 (o superiore) o una chiave ECC.

   > **Note**
   >
   > Per utilizzare un modulo di sicurezza hardware con chiavi ECC, è necessario utilizzare Greengrass nucleus v2.5.6 o versione successiva.
   > Per utilizzare un modulo di sicurezza hardware e un gestore segreto, è necessario utilizzare un modulo di sicurezza hardware con chiavi RSA.

   Consulta la documentazione del tuo HSM per scoprire come inizializzare il token e generare la chiave privata. Se il tuo HSM supporta gli ID degli oggetti, specifica un ID di oggetto quando generi la chiave privata. Salva l'ID dello slot, il PIN utente, l'etichetta dell'oggetto, l'ID dell'oggetto (se il tuo HSM ne utilizza uno) che specifichi quando inizializzi il token e generi la chiave privata. Questi valori vengono utilizzati successivamente quando si importa il certificato dell'oggetto nell'HSM e si configura il AWS IoT Greengrass software Core.
2. Crea una richiesta di firma del certificato (CSR) dalla chiave privata. AWS IoT utilizza questa CSR per creare un thing certificate per la chiave privata generata nell'HSM. Per informazioni su come creare una CSR dalla chiave privata, consulta la documentazione del tuo HSM. La CSR è un file, ad esempio. `iotdevicekey.csr`
3. Copia la CSR dal dispositivo al tuo computer di sviluppo. Se SSH e SCP sono abilitati sul computer di sviluppo e sul dispositivo, puoi usare il `scp` comando sul tuo computer di sviluppo per trasferire la CSR. Sostituisci *`device-ip-address`*con l'indirizzo IP del tuo dispositivo e sostituisci *`~/iotdevicekey.csr con il percorso del file CSR`* sul dispositivo.

```
scp device-ip-address:~/iotdevicekey.csr iotdevicekey.csr
```

4. Sul tuo computer di sviluppo, crea una cartella in cui scaricare il certificato relativo all'oggetto. AWS IoT

```
mkdir greengrass-v2-certs
```

5. Usa il file CSR per creare e scaricare il certificato relativo all' AWS IoT oggetto sul tuo computer di sviluppo.

```
aws iot create-certificate-from-csr --set-as-active --certificate-signing-
request=file://iotdevicekey.csr --certificate-pem-outfile greengrass-v2-certs/
device.pem.crt
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
  "certificateId":
 "aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
  "certificatePem": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
-----END CERTIFICATE-----"
}
```

Salva l'ARN del certificato da utilizzare per configurare il certificato in un secondo momento.

Configura il certificato del oggetto

Allega il thing certificate all' AWS IoT oggetto che hai creato in precedenza e aggiungi una AWS IoT policy al certificato per definire le AWS IoT autorizzazioni per il dispositivo principale.

Per configurare il certificato dell'oggetto

1. Allega il certificato all' AWS IoT oggetto.

   - Sostituiscilo *MyGreengrassCore*con il nome del tuo AWS IoT oggetto.
   - Sostituisci il certificato Amazon Resource Name (ARN) con l'ARN del certificato che hai creato nel passaggio precedente.

   ```
   aws iot attach-thing-principal --thing-name MyGreengrassCore
    --principal arn:aws:iot:us-west-2:123456789012:cert/
   aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
   ```

   Il comando non produce alcun output se la richiesta ha esito positivo.

2. Crea e allega una AWS IoT policy che definisca le AWS IoT autorizzazioni per il tuo dispositivo principale Greengrass. La seguente politica consente l'accesso a tutti gli argomenti MQTT e alle operazioni di Greengrass, in modo che il dispositivo funzioni con applicazioni personalizzate e modifiche future che richiedono nuove operazioni Greengrass. È possibile limitare questa politica in base al proprio caso d'uso. Per ulteriori informazioni, consulta AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali.

   Se hai già configurato un dispositivo Greengrass core, puoi allegare la AWS IoT relativa policy invece di crearne una nuova.

   Esegui questa operazione:

   a. Crea un file che contenga il documento di AWS IoT policy richiesto dai dispositivi core Greengrass.

      Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

      ```
      nano greengrass-v2-iot-policy.json
      ```

Copiate il seguente codice JSON nel file.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "iot:Connect",
        "greengrass:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

b. Crea una AWS IoT politica dal documento di policy.

- Sostituisci *GreenGrassV2IoT ThingPolicy* con il nome della policy da creare.

```
aws iot create-policy --policy-name GreengrassV2IoTThingPolicy --policy-
document file://greengrass-v2-iot-policy.json
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "policyName": "GreengrassV2IoTThingPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
  "policyDocument": "{
    \\"Version\\": \\"2012-10-17\\",
    \\"Statement\\": [
      {
        \\"Effect\\": \\"Allow\\",
        \\"Action\\": [
          \\"iot:Publish\\",
```

```
          \\"iot:Subscribe\\",
          \\"iot:Receive\\",
          \\"iot:Connect\\",
          \\"greengrass:*\\"
        ],
        \\"Resource\\": [
          \\"*\\"
        ]
      }
    ]
  }",
  "policyVersionId": "1"
}
```

c. Allega la AWS IoT policy al certificato dell' AWS IoT oggetto.

- Sostituisci *GreenGrassV2IoT ThingPolicy* con il nome della policy da allegare.
- Sostituisci l'ARN di destinazione con l'ARN del certificato per il tuo oggetto. AWS IoT

```
aws iot attach-policy --policy-name GreengrassV2IoTThingPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

Il comando non ha alcun output se la richiesta ha esito positivo.

Crea un ruolo di scambio di token

I dispositivi core Greengrass utilizzano un ruolo di servizio IAM, chiamato token exchange role, per autorizzare le chiamate ai servizi. AWS Il dispositivo utilizza il provider di AWS IoT credenziali per ottenere AWS credenziali temporanee per questo ruolo, che consente al dispositivo di interagire AWS IoT, inviare log ad Amazon CloudWatch Logs e scaricare elementi dei componenti personalizzati da Amazon S3. Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

Si utilizza un alias di AWS IoT ruolo per configurare il ruolo di scambio di token per i dispositivi principali Greengrass. Gli alias di ruolo consentono di modificare il ruolo di scambio di token per un dispositivo ma mantengono invariata la configurazione del dispositivo. Per ulteriori informazioni, consulta Autorizzazione delle chiamate dirette ai AWS servizi nella Guida per gli AWS IoT Core sviluppatori.

In questa sezione, crei un ruolo IAM per lo scambio di token e un alias di AWS IoT ruolo che rimanda al ruolo. Se hai già configurato un dispositivo principale Greengrass, puoi utilizzare il ruolo di scambio di token e l'alias del ruolo invece di crearne di nuovi. Quindi, configuri il dispositivo in modo che utilizzi quel ruolo e quell'alias. AWS IoT

Per creare un ruolo IAM per lo scambio di token

1. Crea un ruolo IAM che il tuo dispositivo possa utilizzare come ruolo di scambio di token. Esegui questa operazione:

   a. Crea un file che contenga il documento sulla politica di fiducia richiesto dal ruolo di scambio di token.

      Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

      ```
      nano device-role-trust-policy.json
      ```

      Copiate il seguente codice JSON nel file.

      ```
      {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "credentials.iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      }
      ```

   b. Crea il ruolo di scambio di token con il documento sulla politica di fiducia.

      - Sostituisci *greengrassV2 TokenExchangeRole* con il nome del ruolo IAM da creare.

      ```
      aws iam create-role --role-name GreengrassV2TokenExchangeRole --assume-role-
      policy-document file://device-role-trust-policy.json
      ```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "Role": {
    "Path": "/",
    "RoleName": "GreengrassV2TokenExchangeRole",
    "RoleId": "AROAZ2YMUHYHK5OKM77FB",
    "Arn": "arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole",
    "CreateDate": "2021-02-06T00:13:29+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "credentials.iot.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
```

c. Crea un file che contenga il documento sulla politica di accesso richiesto dal ruolo di scambio di token.

Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

```
nano device-role-access-policy.json
```

Copiate il seguente codice JSON nel file.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
```

```
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Questa politica di accesso non consente l'accesso agli artefatti dei componenti nei bucket S3. Per distribuire componenti personalizzati che definiscono gli artefatti in Amazon S3, devi aggiungere autorizzazioni al ruolo per consentire al dispositivo principale di recuperare gli artefatti dei componenti. Per ulteriori informazioni, consulta Consenti l'accesso ai bucket S3 per gli artefatti dei componenti.
> Se non disponi ancora di un bucket S3 per gli artefatti dei componenti, puoi aggiungere queste autorizzazioni in un secondo momento dopo aver creato un bucket.

d. Crea la policy IAM dal documento di policy.

- Sostituisci *GreenGrassV2 TokenExchangeRoleAccess* con il nome della policy IAM da creare.

```
aws iam create-policy --policy-name GreengrassV2TokenExchangeRoleAccess --
policy-document file://device-role-access-policy.json
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "Policy": {
    "PolicyName": "GreengrassV2TokenExchangeRoleAccess",
    "PolicyId": "ANPAZ2YMUHYHACI7C5Z66",
    "Arn": "arn:aws:iam::123456789012:policy/
GreengrassV2TokenExchangeRoleAccess",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
```

```
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2021-02-06T00:37:17+00:00",
    "UpdateDate": "2021-02-06T00:37:17+00:00"
  }
}
```

e. Allega la policy IAM al ruolo di scambio di token.

- Sostituisci *greengrassV2 TokenExchangeRole* con il nome del ruolo IAM.
- Sostituisci l'ARN della policy con l'ARN della policy IAM che hai creato nel passaggio precedente.

```
aws iam attach-role-policy --role-name GreengrassV2TokenExchangeRole --policy-
arn arn:aws:iam::123456789012:policy/GreengrassV2TokenExchangeRoleAccess
```

Il comando non ha alcun output se la richiesta ha esito positivo.

2. Crea un alias di AWS IoT ruolo che punti al ruolo di scambio di token.

- Sostituiscilo *GreengrassCoreTokenExchangeRoleAlias*con il nome dell'alias del ruolo da creare.
- Sostituisci il ruolo ARN con l'ARN del ruolo IAM creato nel passaggio precedente.

```
aws iot create-role-alias --role-alias GreengrassCoreTokenExchangeRoleAlias --role-
arn arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "roleAlias": "GreengrassCoreTokenExchangeRoleAlias",
  "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
}
```

> **Note**
>
> Per creare un alias di ruolo, devi disporre dell'autorizzazione a passare il ruolo IAM per lo scambio di token a. AWS IoT Se ricevi un messaggio di errore quando tenti di creare un alias di ruolo, verifica che AWS l'utente disponga di questa autorizzazione. Per ulteriori informazioni, consulta Concessione a un utente delle autorizzazioni per il trasferimento di un ruolo a un AWS servizio nella Guida per l'AWS Identity and Access Management utente.

3. Crea e allega una AWS IoT policy che consenta al tuo dispositivo principale Greengrass di utilizzare l'alias del ruolo per assumere il ruolo di scambio di token. Se hai già configurato un dispositivo principale Greengrass, puoi allegare la sua AWS IoT politica di alias di ruolo invece di crearne uno nuovo. Esegui questa operazione:

   a. (Facoltativo) Create un file che contenga il documento di AWS IoT policy richiesto dall'alias di ruolo.

      Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

      ```
      nano greengrass-v2-iot-role-alias-policy.json
      ```

      Copiate il seguente codice JSON nel file.

      - Sostituisci l'ARN della risorsa con l'ARN del tuo alias di ruolo.

      ```
      {
        "Version":"2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": "iot:AssumeRoleWithCertificate",
            "Resource": "arn:aws:iot:us-west-2:123456789012:rolealias/
      GreengrassCoreTokenExchangeRoleAlias"
          }
        ]
      }
      ```

b. Crea una AWS IoT politica dal documento di policy.

- Sostituisci *GreengrassCoreTokenExchangeRoleAliasPolicy*con il nome della AWS IoT politica da creare.

```
aws iot create-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --policy-document file://greengrass-v2-iot-role-alias-policy.json
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "policyName": "GreengrassCoreTokenExchangeRoleAliasPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassCoreTokenExchangeRoleAliasPolicy",
  "policyDocument": "{
    \\"Version\\":\\"2012-10-17\\",
    \\"Statement\\": [
      {
        \\"Effect\\": \\"Allow\\",
        \\"Action\\": \\"iot:AssumeRoleWithCertificate\\",
        \\"Resource\\": \\"arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias\\"
      }
    ]
  }",
  "policyVersionId": "1"
}
```

c. Allega la AWS IoT policy al certificato dell' AWS IoT oggetto.

- Sostituisci *GreengrassCoreTokenExchangeRoleAliasPolicy*con il nome della AWS IoT politica relativa agli alias del ruolo.
- Sostituisci l'ARN di destinazione con l'ARN del certificato per il tuo oggetto. AWS IoT

```
aws iot attach-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

Il comando non ha alcun output se la richiesta ha esito positivo.

Scarica i certificati sul dispositivo

In precedenza, avevi scaricato il certificato del dispositivo sul computer di sviluppo. In questa sezione, si copia il certificato sul dispositivo principale per configurare il dispositivo con i certificati a cui viene utilizzato per la connessione AWS IoT. Puoi anche scaricare il certificato Amazon Root Certificate Authority (CA). Se utilizzi un HSM, in questa sezione importi anche il file del certificato nell'HSM.

- Se in precedenza hai creato il certificato e la chiave privata nel AWS IoT servizio, segui i passaggi per scaricare i certificati con chiave privata e file di certificato.
- Se in precedenza hai creato il certificato dell'oggetto da una chiave privata in un modulo di sicurezza hardware (HSM), segui i passaggi per scaricare i certificati con la chiave privata e il certificato in un HSM.

Scarica i certificati con chiave privata e file di certificato

Per scaricare i certificati sul dispositivo

1. Copia il certificato AWS IoT Thing dal tuo computer di sviluppo al dispositivo. Se SSH e SCP sono abilitati sul computer di sviluppo e sul dispositivo, puoi usare il `scp` comando sul tuo computer di sviluppo per trasferire il certificato. *`device-ip-address`*Sostituiscilo con l'indirizzo IP del tuo dispositivo.

   ```
   scp -r greengrass-v2-certs/ device-ip-address:~
   ```

2. Crea la cartella principale Greengrass sul dispositivo. Successivamente installerai il software AWS IoT Greengrass Core in questa cartella.

   Linux or Unix

   - Sostituiscilo */greengrass/v2* con la cartella da usare.

   ```
   sudo mkdir -p /greengrass/v2
   ```

   Windows Command Prompt

   - Sostituire *`C:\greengrass\v2`* con la cartella da utilizzare.

```
mkdir C:\greengrass\v2
```

PowerShell

- Sostituisci *C:\greengrass\v2* con la cartella da usare.

```
mkdir C:\greengrass\v2
```

3. (Solo Linux) Imposta le autorizzazioni del genitore della cartella principale di Greengrass.

   - Sostituisci */greengrass* con il file principale della cartella principale.

   ```
   sudo chmod 755 /greengrass
   ```

4. Copia i AWS IoT Thing Certificates nella cartella principale di Greengrass.

   Linux or Unix

   - Sostituisci */greengrass/v2* con la cartella principale di Greengrass.

   ```
   sudo cp -R ~/greengrass-v2-certs/* /greengrass/v2
   ```

   Windows Command Prompt

   - Sostituisci *C:\greengrass\v2* con la cartella da usare.

   ```
   robocopy %USERPROFILE%\greengrass-v2-certs C:\greengrass\v2 /E
   ```

   PowerShell

   - Sostituisci *C:\greengrass\v2* con la cartella da usare.

   ```
   cp -Path ~\greengrass-v2-certs\* -Destination C:\greengrass\v2
   ```

5. Scarica il certificato Amazon Root Certificate Authority (CA). AWS IoT per impostazione predefinita, i certificati sono associati al certificato CA root di Amazon.

   Linux or Unix

   ```
   sudo curl -o /greengrass/v2/AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   Windows Command Prompt (CMD)

   ```
   curl -o C:\greengrass\v2\\AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   PowerShell

   ```
   iwr -Uri https://www.amazontrust.com/repository/AmazonRootCA1.pem -OutFile C:
   \greengrass\v2\\AmazonRootCA1.pem
   ```

Scarica i certificati con la chiave privata e il certificato in un HSM

> **Note**
>
> Questa funzionalità è disponibile per la versione 2.5.3 e successive del componente Greengrass nucleus. AWS IoT Greengrass attualmente non supporta questa funzionalità sui dispositivi Windows core.

Per scaricare i certificati sul dispositivo

1. Copia il certificato AWS IoT Thing dal tuo computer di sviluppo al dispositivo. Se SSH e SCP sono abilitati sul computer di sviluppo e sul dispositivo, puoi usare il `scp` comando sul tuo computer di sviluppo per trasferire il certificato. *`device-ip-address`*Sostituiscilo con l'indirizzo IP del tuo dispositivo.

   ```
   scp -r greengrass-v2-certs/ device-ip-address:~
   ```

2. Crea la cartella principale Greengrass sul dispositivo. Successivamente installerai il software AWS IoT Greengrass Core in questa cartella.

Linux or Unix

- Sostituiscilo */greengrass/v2* con la cartella da usare.

```
sudo mkdir -p /greengrass/v2
```

Windows Command Prompt

- Sostituire *C:\greengrass\v2* con la cartella da utilizzare.

```
mkdir C:\greengrass\v2
```

PowerShell

- Sostituisci *C:\greengrass\v2* con la cartella da usare.

```
mkdir C:\greengrass\v2
```

3. (Solo Linux) Imposta le autorizzazioni del genitore della cartella principale di Greengrass.

   - Sostituisci */greengrass* con il file principale della cartella principale.

```
sudo chmod 755 /greengrass
```

4. Importa il file del certificato dell'oggetto`~/greengrass-v2-certs/device.pem.crt`, nell'HSM. Consulta la documentazione del tuo HSM per scoprire come importare i certificati al suo interno. Importa il certificato utilizzando lo stesso token, ID slot, PIN utente, etichetta dell'oggetto e ID oggetto (se il tuo HSM ne utilizza uno) con cui hai generato la chiave privata nell'HSM in precedenza.

   > **Note**
   >
   > Se hai generato la chiave privata in precedenza senza un ID di oggetto e il certificato ha un ID oggetto, imposta l'ID dell'oggetto della chiave privata sullo stesso valore del

certificato. Consultate la documentazione del vostro HSM per scoprire come impostare l'ID dell'oggetto per l'oggetto chiave privata.

5. (Facoltativo) Eliminare il file del certificato dell'oggetto, in modo che esista solo nell'HSM.

```
rm ~/greengrass-v2-certs/device.pem.crt
```

6. Scarica il certificato Amazon Root Certificate Authority (CA). AWS IoT per impostazione predefinita, i certificati sono associati al certificato CA root di Amazon.

   Linux or Unix

   ```
   sudo curl -o /greengrass/v2/AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   Windows Command Prompt (CMD)

   ```
   curl -o C:\greengrass\v2\\AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   PowerShell

   ```
   iwr -Uri https://www.amazontrust.com/repository/AmazonRootCA1.pem -OutFile C:
   \greengrass\v2\\AmazonRootCA1.pem
   ```

Configura l'ambiente del dispositivo

Segui i passaggi di questa sezione per configurare un dispositivo Linux o Windows da utilizzare come dispositivo AWS IoT Greengrass principale.

Configura un dispositivo Linux

Per configurare un dispositivo Linux per AWS IoT Greengrass V2

1. Installa il runtime Java, necessario per l'esecuzione del software AWS IoT Greengrass Core. Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva. I seguenti comandi mostrano come installare OpenJDK sul tuo dispositivo.

- Per le distribuzioni basate su Debian o basate su Ubuntuu:

```
sudo apt install default-jdk
```

- Per le distribuzioni basate su Red Hat:

```
sudo yum install java-11-openjdk-devel
```

- Per Amazon Linux 2:

```
sudo amazon-linux-extras install java-openjdk11
```

- Per Amazon Linux 2023:

```
sudo dnf install java-11-amazon-corretto -y
```

Al termine dell'installazione, esegui il comando seguente per verificare che Java sia in esecuzione sul tuo dispositivo Linux.

```
java -version
```

Il comando stampa la versione di Java in esecuzione sul dispositivo. Ad esempio, su una distribuzione basata su Debian, l'output potrebbe essere simile all'esempio seguente.

```
openjdk version "11.0.9.1" 2020-11-04
OpenJDK Runtime Environment (build 11.0.9.1+1-post-Debian-1deb10u2)
OpenJDK 64-Bit Server VM (build 11.0.9.1+1-post-Debian-1deb10u2, mixed mode)
```

2. (Facoltativo) Crea l'utente e il gruppo di sistema predefiniti che eseguono i componenti sul dispositivo. Puoi anche scegliere di lasciare che il programma di installazione del software AWS IoT Greengrass Core crei questo utente e gruppo durante l'installazione con l'argomento `--component-default-user` installer. Per ulteriori informazioni, consulta Argomenti dell'installatore.

```
sudo useradd --system --create-home ggc_user
sudo groupadd --system ggc_group
```

3. Verificate che l'utente che esegue il software AWS IoT Greengrass Core (in genere`root`) sia autorizzato a funzionare `sudo` con qualsiasi utente e gruppo.

   a. Eseguite il comando seguente per aprire il `/etc/sudoers` file.

   ```
   sudo visudo
   ```

   b. Verificate che l'autorizzazione per l'utente sia simile all'esempio seguente.

   ```
   root    ALL=(ALL:ALL) ALL
   ```

4. (Facoltativo) Per eseguire funzioni Lambda containerizzate, è necessario abilitare cgroups v1 e abilitare e montare i cgroup di memoria e dispositivi. Se non intendi eseguire funzioni Lambda containerizzate, puoi saltare questo passaggio.

   Per abilitare queste opzioni di cgroups, avvia il dispositivo con i seguenti parametri del kernel Linux.

   ```
   cgroup_enable=memory cgroup_memory=1 systemd.unified_cgroup_hierarchy=0
   ```

   Per informazioni sulla visualizzazione e l'impostazione dei parametri del kernel per il tuo dispositivo, consulta la documentazione del tuo sistema operativo e del boot loader. Segui le istruzioni per impostare in modo permanente i parametri del kernel.

5. Installa tutte le altre dipendenze richieste sul tuo dispositivo come indicato dall'elenco dei requisiti in. Requisiti per il dispositivo

Configura un dispositivo Windows

> **Note**
>
> Questa funzionalità è disponibile per la versione 2.5.0 e successive del componente Greengrass nucleus.

Per configurare un dispositivo Windows per AWS IoT Greengrass V2

1. Installa il runtime Java, necessario per l'esecuzione del software AWS IoT Greengrass Core. Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva.

2. Controlla se Java è disponibile nella variabile di sistema PATH e aggiungilo in caso contrario. L' LocalSystem account esegue il software AWS IoT Greengrass Core, quindi è necessario aggiungere Java alla variabile di sistema PATH anziché alla variabile utente PATH per l'utente. Esegui questa operazione:

   a. Premi il tasto Windows per aprire il menu di avvio.

   b. Digita **`environment variables`** per cercare le opzioni di sistema dal menu di avvio.

   c. Nei risultati della ricerca del menu di avvio, scegli Modifica le variabili di ambiente di sistema per aprire la finestra delle proprietà del sistema.

   d. Scegli le variabili di ambiente... per aprire la finestra Variabili d'ambiente.

   e. In Variabili di sistema, seleziona Percorso, quindi scegli Modifica. Nella finestra Modifica variabile di ambiente, puoi visualizzare ogni percorso su una riga separata.

   f. Controlla se è presente il percorso della `bin` cartella di installazione di Java. Il percorso potrebbe essere simile all'esempio seguente.

   ```
   C:\\Program Files\\Amazon Corretto\\jdk11.0.13_8\\bin
   ```

   g. Se la `bin` cartella di installazione Java non è presente in Path, scegliete Nuovo per aggiungerla, quindi scegliete OK.

3. Aprite il prompt dei comandi di Windows (`cmd.exe`) come amministratore.

4. Crea l'utente predefinito nell' LocalSystem account sul dispositivo Windows. Sostituisci *`la password`* con una password sicura.

   ```
   net user /add ggc_user password
   ```

   > **Tip**
   >
   > A seconda della configurazione di Windows, la password dell'utente potrebbe essere impostata per scadere in date future. Per garantire che le tue applicazioni Greengrass continuino a funzionare, tieni traccia della scadenza della password e aggiornala prima che scada. Puoi anche impostare la password dell'utente in modo che non scada mai.
   >
   > - Per verificare la scadenza di un utente e della relativa password, esegui il comando seguente.
   >
   >   ```
   >   net user ggc_user | findstr /C:expires
   >   ```

- Per impostare la password di un utente in modo che non scada mai, esegui il comando seguente.

  ```
  wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
  ```

- Se utilizzi Windows 10 o versioni successive in cui il wmiccomando è obsoleto, esegui il comando seguente. PowerShell

  ```
  Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
   'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
  ```

5. Scarica e installa l'PsExecutilità di Microsoft sul dispositivo.
6. Utilizzate l' PsExec utilità per memorizzare il nome utente e la password per l'utente predefinito nell'istanza di Credential Manager per l' LocalSystem account. Sostituisci la *password* con la password dell'utente impostata in precedenza.

   ```
   psexec -s cmd /c cmdkey /generic:ggc_user /user:ggc_user /pass:password
   ```

   Se si PsExec License Agreementapre, scegli Acceptdi accettare la licenza ed esegui il comando.

   > **Note**
   >
   > Sui dispositivi Windows, l' LocalSystem account esegue il Greengrass nucleus ed è necessario utilizzare l' PsExec utilità per memorizzare le informazioni utente predefinite nell'account. LocalSystem L'utilizzo dell'applicazione Credential Manager archivia queste informazioni nell'account Windows dell'utente attualmente connesso, anziché nell'account. LocalSystem

Scaricate il software Core AWS IoT Greengrass

È possibile scaricare la versione più recente del software AWS IoT Greengrass Core dal seguente indirizzo:

- https://d2s8p88vqu9w66.cloudfront.net/releases/ greengrass-nucleus-latest .zip

> **Note**
>
> È possibile scaricare una versione specifica del software AWS IoT Greengrass Core dal seguente percorso. Sostituisci la *versione* con la versione da scaricare.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Per scaricare il software AWS IoT Greengrass Core

1. Sul dispositivo principale, scaricate il software AWS IoT Greengrass Core in un file denominato`greengrass-nucleus-latest.zip`.

   Linux or Unix

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
   latest.zip > greengrass-nucleus-latest.zip
   ```

   Windows Command Prompt (CMD)

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
   latest.zip > greengrass-nucleus-latest.zip
   ```

   PowerShell

   ```
   iwr -Uri https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
   latest.zip -OutFile greengrass-nucleus-latest.zip
   ```

   Scaricando questo software accetti l'Accordo di licenza del software Greengrass Core.

2. (Facoltativo) Per verificare la firma del software Greengrass nucleus

   > **Note**
   >
   > Questa funzionalità è disponibile con Greengrass nucleus versione 2.9.5 e successive.

   a. Usa il seguente comando per verificare la firma del tuo artefatto Greengrass nucleus:

Linux or Unix

```
jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
```

Windows Command Prompt (CMD)

Il nome del file potrebbe avere un aspetto diverso a seconda della versione di JDK installata. *jdk17.0.6_10*Sostituiscilo con la versione JDK che hai installato.

```
"C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
verify -certs -verbose greengrass-nucleus-latest.zip
```

PowerShell

Il nome del file potrebbe avere un aspetto diverso a seconda della versione di JDK installata. *jdk17.0.6_10*Sostituiscilo con la versione JDK che hai installato.

```
'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
verify -certs -verbose greengrass-nucleus-latest.zip
```

b. L'`jarsigner`invocazione produce un output che indica i risultati della verifica.

   i. Se il file zip Greengrass nucleus è firmato, l'output contiene la seguente dichiarazione:

   ```
   jar verified.
   ```

   ii. Se il file zip Greengrass nucleus non è firmato, l'output contiene la seguente dichiarazione:

   ```
   jar is unsigned.
   ```

c. Se hai fornito l'`-certs`opzione Jarsigner insieme alle `-verbose` opzioni `-verify` e, l'output include anche informazioni dettagliate sul certificato del firmatario.

3. Decomprimi il software AWS IoT Greengrass Core in una cartella sul tuo dispositivo. Sostituiscilo *GreengrassInstaller*con la cartella che desideri utilizzare.

Linux or Unix

```
unzip greengrass-nucleus-latest.zip -d GreengrassInstaller && rm greengrass-
nucleus-latest.zip
```

Windows Command Prompt (CMD)

```
mkdir GreengrassInstaller && tar -xf greengrass-nucleus-latest.zip -
C GreengrassInstaller && del greengrass-nucleus-latest.zip
```

PowerShell

```
Expand-Archive -Path greengrass-nucleus-latest.zip -DestinationPath .\
\GreengrassInstaller
rm greengrass-nucleus-latest.zip
```

4. (Facoltativo) Eseguite il comando seguente per visualizzare la versione del software AWS IoT Greengrass Core.

```
java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
```

> **Important**
>
> Se installi una versione del nucleo Greengrass precedente alla v2.4.0, non rimuovere questa cartella dopo aver installato il software Core. AWS IoT Greengrass Il software AWS IoT Greengrass Core utilizza i file in questa cartella per l'esecuzione.
> Se hai scaricato la versione più recente del software, installi la versione 2.4.0 o successiva e puoi rimuovere questa cartella dopo aver installato il software AWS IoT Greengrass Core.

Installa il software Core AWS IoT Greengrass

Esegui il programma di installazione con argomenti che specificano le seguenti azioni:

- Esegui l'installazione da un file di configurazione parziale che specifica di utilizzare le AWS risorse e i certificati creati in precedenza. Il software AWS IoT Greengrass Core utilizza un file di configurazione che specifica la configurazione di ogni componente Greengrass sul

dispositivo. Il programma di installazione crea un file di configurazione completo a partire dal file di configurazione parziale fornito dall'utente.

- Specificate di utilizzare l'utente `ggc_user` del sistema per eseguire i componenti software sul dispositivo principale. Sui dispositivi Linux, questo comando specifica anche di utilizzare il gruppo di `ggc_group` sistema e il programma di installazione crea automaticamente l'utente e il gruppo di sistema.
- Configura il software AWS IoT Greengrass Core come servizio di sistema che viene eseguito all'avvio. Sui dispositivi Linux, ciò richiede il sistema di inizializzazione Systemd.

> **Important**
>
> Sui dispositivi Windows core, è necessario configurare il software AWS IoT Greengrass Core come servizio di sistema.

Per ulteriori informazioni sugli argomenti che è possibile specificare, vedereArgomenti dell'installatore.

> **Note**
>
> Se utilizzi AWS IoT Greengrass un dispositivo con memoria limitata, puoi controllare la quantità di memoria utilizzata dal software AWS IoT Greengrass Core. Per controllare l'allocazione della memoria, è possibile impostare le opzioni relative alla dimensione dell'heap JVM nel parametro di `jvmOptions` configurazione del componente nucleus. Per ulteriori informazioni, consulta Controlla l'allocazione della memoria con le opzioni JVM.

- Se in precedenza hai creato il certificato e la chiave privata nel AWS IoT servizio, segui i passaggi per installare il software AWS IoT Greengrass Core con la chiave privata e i file di certificato.
- Se in precedenza hai creato il certificato dell'oggetto da una chiave privata in un modulo di sicurezza hardware (HSM), segui i passaggi per installare il software AWS IoT Greengrass Core con la chiave privata e il certificato in un HSM.

Installa il software AWS IoT Greengrass Core con chiave privata e file di certificato

Per installare il software AWS IoT Greengrass Core

1. Controlla la versione del software AWS IoT Greengrass Core.

   - Sostituisci *GreengrassInstaller* con il percorso della cartella che contiene il software.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

2. Utilizzate un editor di testo per creare un file di configurazione denominato `config.yaml` da fornire all'installatore.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

   ```
   nano GreengrassInstaller/config.yaml
   ```

   Copiate il seguente contenuto YAML nel file. Questo file di configurazione parziale specifica i parametri di sistema e i parametri del nucleo di Greengrass.

   ```
   ---
   system:
     certificateFilePath: "/greengrass/v2/device.pem.crt"
     privateKeyPath: "/greengrass/v2/private.pem.key"
     rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
     rootpath: "/greengrass/v2"
     thingName: "MyGreengrassCore"
   services:
     aws.greengrass.Nucleus:
       componentType: "NUCLEUS"
       version: "2.12.3"
       configuration:
         awsRegion: "us-west-2"
         iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
         iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
         iotCredEndpoint: "device-credentials-prefix.credentials.iot.us-
   west-2.amazonaws.com"
   ```

   Successivamente, esegui queste operazioni:

- Sostituisci ogni istanza di */greengrass/v2* con la cartella principale di Greengrass.
- Sostituisci *MyGreengrassCore*con il nome della AWS IoT cosa.
- Sostituire *2.12.3* con la versione del software AWS IoT Greengrass Core.
- Sostituisci *us-west-2* con Regione AWS il luogo in cui hai creato le risorse.
- Sostituisci *GreengrassCoreTokenExchangeRoleAlias*con il nome dell'alias del ruolo di scambio di token.
- Sostituiscilo `iotDataEndpoint` con il tuo endpoint di AWS IoT dati.
- Sostituisci l'`iotCredEndpoint`endpoint con AWS IoT le tue credenziali.

> **Note**
>
> In questo file di configurazione, è possibile personalizzare altre opzioni di configurazione del nucleo, come le porte e il proxy di rete da utilizzare, come mostrato nell'esempio seguente. Per ulteriori informazioni, vedere Greengrass nucleus configuration.

```
---
system:
  certificateFilePath: "/greengrass/v2/device.pem.crt"
  privateKeyPath: "/greengrass/v2/private.pem.key"
  rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
  rootpath: "/greengrass/v2"
  thingName: "MyGreengrassCore"
services:
  aws.greengrass.Nucleus:
    componentType: "NUCLEUS"
    version: "2.12.3"
    configuration:
      awsRegion: "us-west-2"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      iotCredEndpoint: "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
      mqtt:
        port: 443
      greengrassDataPlanePort: 443
      networkProxy:
        noProxyAddresses: "http://192.168.0.1,www.example.com"
        proxy:
```

```
            url: "https://my-proxy-server:1100"
            username: "Mary_Major"
            password: "pass@word1357"
```

3. Eseguite il programma di installazione e specificate di `--init-config` fornire il file di configurazione.

   - Sostituire */greengrass/v2* o *C:\greengrass\v2* con la cartella principale di Greengrass.
   - Sostituisci ogni istanza di *GreengrassInstaller*con la cartella in cui hai decompresso il programma di installazione.

   Linux or Unix

   ```
   sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
     -jar ./GreengrassInstaller/lib/Greengrass.jar \
     --init-config ./GreengrassInstaller/config.yaml \
     --component-default-user ggc_user:ggc_group \
     --setup-system-service true
   ```

   Windows Command Prompt (CMD)

   ```
   java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" ^
     -jar ./GreengrassInstaller/lib/Greengrass.jar ^
     --init-config ./GreengrassInstaller/config.yaml ^
     --component-default-user ggc_user ^
     --setup-system-service true
   ```

   PowerShell

   ```
   java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" `
     -jar ./GreengrassInstaller/lib/Greengrass.jar `
     --init-config ./GreengrassInstaller/config.yaml `
     --component-default-user ggc_user `
     --setup-system-service true
   ```

> **Important**
>
> Nei dispositivi Windows Core, è necessario specificare `--setup-system-service true` di configurare il software AWS IoT Greengrass Core come servizio di sistema.

Se si specifica`--setup-system-service true`, il programma di installazione stampa `Successfully set up Nucleus as a system service` se ha configurato ed eseguito il software come servizio di sistema. In caso contrario, il programma di installazione non emette alcun messaggio se installa il software correttamente.

> **Note**
>
> Non è possibile utilizzare l'`deploy-dev-tools`argomento per distribuire strumenti di sviluppo locali quando si esegue il programma di installazione senza l'argomento. `--provision true` Per informazioni sulla distribuzione della CLI Greengrass direttamente sul dispositivo, consulta. Interfaccia a riga di comando Greengrass

4. Verifica l'installazione visualizzando i file nella cartella principale.

   Linux or Unix

   ```
   ls /greengrass/v2
   ```

   Windows Command Prompt (CMD)

   ```
   dir C:\greengrass\v2
   ```

   PowerShell

   ```
   ls C:\greengrass\v2
   ```

   Se l'installazione è riuscita, la cartella principale contiene diverse cartelle, ad esempio `configpackages`, `elogs`.

Installa il software AWS IoT Greengrass Core con la chiave privata e il certificato in un HSM

> **Note**
>
> Questa funzionalità è disponibile per la versione 2.5.3 e successive del componente Greengrass nucleus. AWS IoT Greengrass attualmente non supporta questa funzionalità sui dispositivi Windows core.

Per installare il software AWS IoT Greengrass Core

1. Controlla la versione del software AWS IoT Greengrass Core.

   - Sostituisci *GreengrassInstaller* con il percorso della cartella che contiene il software.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

2. Per consentire al software AWS IoT Greengrass Core di utilizzare la chiave privata e il certificato nell'HSM, installate il componente del provider PKCS #11 quando installate il software AWS IoT Greengrass Core. Il componente del provider PKCS #11 è un plug-in che è possibile configurare durante l'installazione. È possibile scaricare la versione più recente del componente del provider PKCS #11 dalla seguente posizione:

   - https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/aws.greengrass.crypto.Pkcs11Provider-latest.jar

   Scarica il plugin del provider PKCS #11 in un file denominato `aws.greengrass.crypto.Pkcs11Provider.jar`. Sostituiscilo *GreengrassInstaller* con la cartella che desideri utilizzare.

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/
   aws.greengrass.crypto.Pkcs11Provider-latest.jar > GreengrassInstaller/
   aws.greengrass.crypto.Pkcs11Provider.jar
   ```

   Scaricando questo software accetti l'Accordo di licenza del software Greengrass Core.

3. Utilizzate un editor di testo per creare un file di configurazione denominato `config.yaml` da fornire all'installatore.

Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

```
nano GreengrassInstaller/config.yaml
```

Copiate il seguente contenuto YAML nel file. Questo file di configurazione parziale specifica i parametri di sistema, i parametri Greengrass nucleus e i parametri del provider PKCS #11.

```
---
system:
  certificateFilePath: "pkcs11:object=iotdevicekey;type=cert"
  privateKeyPath: "pkcs11:object=iotdevicekey;type=private"
  rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
  rootpath: "/greengrass/v2"
  thingName: "MyGreengrassCore"
services:
  aws.greengrass.Nucleus:
    componentType: "NUCLEUS"
    version: "2.12.3"
    configuration:
      awsRegion: "us-west-2"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
      iotCredEndpoint: "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
  aws.greengrass.crypto.Pkcs11Provider:
    configuration:
      name: "softhsm_pkcs11"
      library: "/usr/local/Cellar/softhsm/2.6.1/lib/softhsm/libsofthsm2.so"
      slot: 1
      userPin: "1234"
```

Successivamente, esegui queste operazioni:

- Sostituisci ogni istanza di *iotdevicekey* negli URI PKCS #11 con l'etichetta dell'oggetto in cui hai creato la chiave privata e importato il certificato.
- Sostituisci ogni istanza di */greengrass/v2* con la cartella principale di Greengrass.
- Sostituisci *MyGreengrassCore*con il nome della AWS IoT cosa.
- Sostituire *2.12.3* con la versione del software AWS IoT Greengrass Core.

- Sostituisci *us-west-2* con Regione AWS il luogo in cui hai creato le risorse.
- Sostituisci *GreengrassCoreTokenExchangeRoleAlias*con il nome dell'alias del ruolo di scambio di token.
- Sostituiscilo `iotDataEndpoint` con il tuo endpoint di AWS IoT dati.
- Sostituisci l'`iotCredEndpoint`endpoint con AWS IoT le tue credenziali.
- Sostituisci i parametri di configurazione per il `aws.greengrass.crypto.Pkcs11Provider` componente con i valori per la configurazione HSM sul dispositivo principale.

> **Note**
>
> In questo file di configurazione, è possibile personalizzare altre opzioni di configurazione del nucleo, come le porte e il proxy di rete da utilizzare, come mostrato nell'esempio seguente. Per ulteriori informazioni, vedere Greengrass nucleus configuration.

```
---
system:
  certificateFilePath: "pkcs11:object=iotdevicekey;type=cert"
  privateKeyPath: "pkcs11:object=iotdevicekey;type=private"
  rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
  rootpath: "/greengrass/v2"
  thingName: "MyGreengrassCore"
services:
  aws.greengrass.Nucleus:
    componentType: "NUCLEUS"
    version: "2.12.3"
    configuration:
      awsRegion: "us-west-2"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
      iotCredEndpoint: "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
      mqtt:
        port: 443
      greengrassDataPlanePort: 443
      networkProxy:
        noProxyAddresses: "http://192.168.0.1,www.example.com"
        proxy:
          url: "https://my-proxy-server:1100"
          username: "Mary_Major"
```

```
          password: "pass@word1357"
    aws.greengrass.crypto.Pkcs11Provider:
      configuration:
        name: "softhsm_pkcs11"
        library: "/usr/local/Cellar/softhsm/2.6.1/lib/softhsm/libsofthsm2.so"
        slot: 1
        userPin: "1234"
```

4. Eseguite il programma di installazione e specificate di `--init-config` fornire il file di configurazione.

   - Sostituisci */greengrass/v2* con la cartella principale di Greengrass.
   - Sostituisci ogni istanza di *GreengrassInstaller*con la cartella in cui hai decompresso il programma di installazione.

   ```
   sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
     -jar ./GreengrassInstaller/lib/Greengrass.jar \
     --trusted-plugin ./GreengrassInstaller/aws.greengrass.crypto.Pkcs11Provider.jar \
     --init-config ./GreengrassInstaller/config.yaml \
     --component-default-user ggc_user:ggc_group \
     --setup-system-service true
   ```

   Important

   Nei dispositivi Windows Core, è necessario specificare `--setup-system-service true` di configurare il software AWS IoT Greengrass Core come servizio di sistema.

   Se si specifica`--setup-system-service true`, il programma di installazione stampa `Successfully set up Nucleus as a system service` se ha configurato ed eseguito il software come servizio di sistema. In caso contrario, il programma di installazione non emette alcun messaggio se installa il software correttamente.

   Note

   Non è possibile utilizzare l'`deploy-dev-tools`argomento per distribuire strumenti di sviluppo locali quando si esegue il programma di installazione senza l'argomento.

`--provision true` Per informazioni sulla distribuzione della CLI Greengrass direttamente sul dispositivo, consulta. Interfaccia a riga di comando Greengrass

5. Verifica l'installazione visualizzando i file nella cartella principale.

   Linux or Unix

   ```
   ls /greengrass/v2
   ```

   Windows Command Prompt (CMD)

   ```
   dir C:\greengrass\v2
   ```

   PowerShell

   ```
   ls C:\greengrass\v2
   ```

   Se l'installazione è riuscita, la cartella principale contiene diverse cartelle, ad esempio `configpackages`, `elogs`.

Se avete installato il software AWS IoT Greengrass Core come servizio di sistema, il programma di installazione esegue il software automaticamente. In caso contrario, è necessario eseguire il software manualmente. Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core.

Per ulteriori informazioni su come configurare e utilizzare il software AWS IoT Greengrass, consulta quanto segue:

- Configurare il software AWS IoT Greengrass Core
- Sviluppa AWS IoT Greengrass componenti
- Implementazione AWS IoT Greengrass dei componenti sui dispositivi
- Interfaccia a riga di comando Greengrass

Installa il software AWS IoT Greengrass Core con il provisioning AWS IoT della flotta

Questa funzionalità è disponibile per la versione 2.4.0 e successive del componente Greengrass nucleus.

Con AWS IoT Fleet Provisioning, è possibile AWS IoT configurare la generazione e la distribuzione sicura di certificati e chiavi private X.509 ai dispositivi quando si connettono per la prima volta. AWS IoT AWS IoT fornisce certificati client firmati dall'autorità di certificazione Amazon Root (CA). Puoi anche configurare AWS IoT per specificare gruppi di oggetti, tipi di oggetti e autorizzazioni per i dispositivi core Greengrass di cui effettui il provisioning con il fleet provisioning. È possibile definire un modello di provisioning per definire il modo AWS IoT in cui viene effettuato il provisioning di ciascun dispositivo. Il modello di provisioning specifica l'oggetto, la policy e le risorse dei certificati da creare per un dispositivo durante il provisioning. Per ulteriori informazioni, consulta Provisioning templates nella Developer Guide.AWS IoT Core

AWS IoT Greengrass fornisce un plug-in per il provisioning del AWS IoT parco veicoli che è possibile utilizzare per installare il software AWS IoT Greengrass Core utilizzando AWS le risorse create dal AWS IoT fleet provisioning. Il plug-in per il provisioning della flotta utilizza il provisioning by claim. I dispositivi utilizzano un certificato di richiesta di provisioning e una chiave privata per ottenere un certificato e una chiave privata univoci del dispositivo X.509 da utilizzare per le normali operazioni. È possibile incorporare il certificato di richiesta e la chiave privata in ogni dispositivo durante la produzione, in modo che i clienti possano attivare i dispositivi in un secondo momento, quando ogni dispositivo sarà online. Puoi utilizzare lo stesso certificato di richiesta e la stessa chiave privata per più dispositivi. Per ulteriori informazioni, consulta Provisioning by claim nella AWS IoT Core Developer Guide.

> **Note**
>
> Il plug-in fleet provisioning attualmente non supporta la memorizzazione di chiavi private e file di certificato in un modulo di sicurezza hardware (HSM). Per utilizzare un HSM, installa il software AWS IoT Greengrass Core con provisioning manuale.

Per installare il software AWS IoT Greengrass Core con il provisioning AWS IoT della flotta, è necessario configurare le risorse nel dispositivo AWS IoT utilizzato per Account AWS il provisioning dei dispositivi core Greengrass. Queste risorse includono un modello di provisioning, certificati di richiesta e un ruolo IAM per lo scambio di token. Dopo aver creato queste risorse, puoi riutilizzarle

per effettuare il provisioning di più dispositivi principali in una flotta. Per ulteriori informazioni, consulta Configura il provisioning AWS IoT della flotta per i dispositivi core Greengrass.

> **Important**
>
> Prima di scaricare il software AWS IoT Greengrass Core, verificate che il dispositivo principale soddisfi i requisiti per installare ed eseguire il software AWS IoT Greengrass Core v2.0.

Argomenti

Prerequisiti

Per installare il software AWS IoT Greengrass Core con il provisioning AWS IoT della flotta, devi prima configurare il provisioning del AWS IoT parco veicoli per i dispositivi core Greengrass. Dopo aver completato questi passaggi una volta, puoi utilizzare il provisioning del parco veicoli per installare il software AWS IoT Greengrass Core su un numero qualsiasi di dispositivi.

Recupera AWS IoT gli endpoint

Ottieni gli AWS IoT endpoint per te e salvali per Account AWS utilizzarli in un secondo momento. Il tuo dispositivo utilizza questi endpoint per connettersi a. AWS IoT Esegui questa operazione:

1. Ottieni l'endpoint di AWS IoT dati per il tuo. Account AWS

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
}
```

2. Ottieni l'endpoint delle AWS IoT credenziali per il tuo. Account AWS

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "endpointAddress": "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
}
```

Scarica i certificati sul dispositivo

Il dispositivo utilizza un certificato di richiesta e una chiave privata per autenticare la richiesta di fornitura di AWS risorse e acquisire un certificato del dispositivo X.509. È possibile incorporare il certificato di richiesta e la chiave privata nel dispositivo durante la produzione oppure copiare il certificato e la chiave sul dispositivo durante l'installazione. In questa sezione, copi il certificato di richiesta e la chiave privata sul dispositivo. Puoi anche scaricare il certificato Amazon Root Certificate Authority (CA) sul dispositivo.

> **Important**
>
> Il provisioning dichiara che le chiavi private devono essere protette in ogni momento, anche sui dispositivi core Greengrass. Ti consigliamo di utilizzare i CloudWatch parametri e i log di Amazon per monitorare eventuali indicazioni di uso improprio, come l'uso non autorizzato del certificato di attestazione per il provisioning dei dispositivi. Se rilevi un uso improprio, disattiva il certificato di richiesta di approvvigionamento in modo che non possa essere utilizzato per il provisioning dei dispositivi. Per ulteriori informazioni, consulta Monitoring AWS IoT nella Developer Guide.AWS IoT Core

Per aiutarti a gestire meglio il numero di dispositivi e i dispositivi che si registrano automaticamente nel tuo sistema Account AWS, puoi specificare un hook di pre-provisioning quando crei un modello di provisioning del parco veicoli. Un hook di pre-provisioning è una AWS Lambda funzione che convalida i parametri del modello forniti dai dispositivi durante la registrazione. Ad esempio, è possibile creare un hook di pre-provisioning che controlli l'ID di un dispositivo confrontandolo con un database per verificare che il dispositivo disponga dell'autorizzazione al provisioning. Per ulteriori informazioni, consulta Pre-provisioning hook nella Developer Guide.AWS IoT Core

Per scaricare i certificati di richiesta sul dispositivo

1. Copia il certificato di richiesta e la chiave privata sul dispositivo. Se SSH e SCP sono abilitati sul computer di sviluppo e sul dispositivo, puoi utilizzare il `scp` comando sul tuo computer di sviluppo per trasferire il certificato di richiesta e la chiave privata. Il comando di esempio seguente trasferisce questi file in una cartella denominata `claim-certs` sul computer di sviluppo al dispositivo. Sostituiscilo *`device-ip-address`*con l'indirizzo IP del tuo dispositivo.

   ```
   scp -r claim-certs/ device-ip-address:~
   ```

2. Crea la cartella principale Greengrass sul dispositivo. Successivamente installerai il software AWS IoT Greengrass Core in questa cartella.

   Linux or Unix

   - Sostituiscilo */greengrass/v2* con la cartella da usare.

   ```
   sudo mkdir -p /greengrass/v2
   ```

   Windows Command Prompt

   - Sostituire *C:\greengrass\v2* con la cartella da utilizzare.

   ```
   mkdir C:\greengrass\v2
   ```

PowerShell

- Sostituisci *C:\greengrass\v2* con la cartella da usare.

```
mkdir C:\greengrass\v2
```

3. (Solo Linux) Imposta le autorizzazioni del genitore della cartella principale di Greengrass.

   - Sostituisci */greengrass* con il file principale della cartella principale.

```
sudo chmod 755 /greengrass
```

4. Sposta i certificati di reclamo nella cartella principale di Greengrass.

   - Sostituire */greengrass/v2* o *C:\greengrass\v2* con la cartella principale di Greengrass.

   Linux or Unix

```
sudo mv ~/claim-certs /greengrass/v2
```

   Windows Command Prompt (CMD)

```
move %USERPROFILE%\claim-certs C:\greengrass\v2
```

   PowerShell

```
mv -Path ~\claim-certs -Destination C:\greengrass\v2
```

5. Scarica il certificato Amazon Root Certificate Authority (CA). AWS IoT per impostazione predefinita, i certificati sono associati al certificato CA root di Amazon.

   Linux or Unix

```
sudo curl -o /greengrass/v2/AmazonRootCA1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem
```

Windows Command Prompt (CMD)

```
curl -o C:\greengrass\v2\\AmazonRootCA1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem
```

PowerShell

```
iwr -Uri https://www.amazontrust.com/repository/AmazonRootCA1.pem -OutFile C:
\greengrass\v2\\AmazonRootCA1.pem
```

Configura l'ambiente del dispositivo

Segui i passaggi di questa sezione per configurare un dispositivo Linux o Windows da utilizzare come dispositivo AWS IoT Greengrass principale.

Configura un dispositivo Linux

Per configurare un dispositivo Linux per AWS IoT Greengrass V2

1. Installa il runtime Java, necessario per l'esecuzione del software AWS IoT Greengrass Core. Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva. I seguenti comandi mostrano come installare OpenJDK sul tuo dispositivo.

   - Per le distribuzioni basate su Debian o basate su Ubuntuu:

     ```
     sudo apt install default-jdk
     ```

   - Per le distribuzioni basate su Red Hat:

     ```
     sudo yum install java-11-openjdk-devel
     ```

   - Per Amazon Linux 2:

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - Per Amazon Linux 2023:

     ```
     sudo dnf install java-11-amazon-corretto -y
     ```

Al termine dell'installazione, esegui il comando seguente per verificare che Java sia in esecuzione sul tuo dispositivo Linux.

```
java -version
```

Il comando stampa la versione di Java in esecuzione sul dispositivo. Ad esempio, su una distribuzione basata su Debian, l'output potrebbe essere simile all'esempio seguente.

```
openjdk version "11.0.9.1" 2020-11-04
OpenJDK Runtime Environment (build 11.0.9.1+1-post-Debian-1deb10u2)
OpenJDK 64-Bit Server VM (build 11.0.9.1+1-post-Debian-1deb10u2, mixed mode)
```

2. (Facoltativo) Crea l'utente e il gruppo di sistema predefiniti che eseguono i componenti sul dispositivo. Puoi anche scegliere di lasciare che il programma di installazione del software AWS IoT Greengrass Core crei questo utente e gruppo durante l'installazione con l'argomento `--component-default-user` installer. Per ulteriori informazioni, consulta Argomenti dell'installatore.

```
sudo useradd --system --create-home ggc_user
sudo groupadd --system ggc_group
```

3. Verificate che l'utente che esegue il software AWS IoT Greengrass Core (in genere`root`) sia autorizzato a funzionare `sudo` con qualsiasi utente e gruppo.

   a. Eseguite il comando seguente per aprire il `/etc/sudoers` file.

   ```
   sudo visudo
   ```

   b. Verificate che l'autorizzazione per l'utente sia simile all'esempio seguente.

   ```
   root    ALL=(ALL:ALL) ALL
   ```

4. (Facoltativo) Per eseguire funzioni Lambda containerizzate, è necessario abilitare cgroups v1 e abilitare e montare i cgroup di memoria e dispositivi. Se non intendi eseguire funzioni Lambda containerizzate, puoi saltare questo passaggio.

   Per abilitare queste opzioni di cgroups, avvia il dispositivo con i seguenti parametri del kernel Linux.

```
cgroup_enable=memory cgroup_memory=1 systemd.unified_cgroup_hierarchy=0
```

Per informazioni sulla visualizzazione e l'impostazione dei parametri del kernel per il tuo dispositivo, consulta la documentazione del tuo sistema operativo e del boot loader. Segui le istruzioni per impostare in modo permanente i parametri del kernel.

5. Installa tutte le altre dipendenze richieste sul tuo dispositivo come indicato dall'elenco dei requisiti in. Requisiti per il dispositivo

Configura un dispositivo Windows

> **Note**
>
> Questa funzionalità è disponibile per la versione 2.5.0 e successive del componente Greengrass nucleus.

Per configurare un dispositivo Windows per AWS IoT Greengrass V2

1. Installa il runtime Java, necessario per l'esecuzione del software AWS IoT Greengrass Core. Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva.
2. Controlla se Java è disponibile nella variabile di sistema PATH e aggiungilo in caso contrario. L' LocalSystem account esegue il software AWS IoT Greengrass Core, quindi è necessario aggiungere Java alla variabile di sistema PATH anziché alla variabile utente PATH per l'utente. Esegui questa operazione:

   a. Premi il tasto Windows per aprire il menu di avvio.
   b. Digita **`environment variables`** per cercare le opzioni di sistema dal menu di avvio.
   c. Nei risultati della ricerca del menu di avvio, scegli Modifica le variabili di ambiente di sistema per aprire la finestra delle proprietà del sistema.
   d. Scegli le variabili di ambiente... per aprire la finestra Variabili d'ambiente.
   e. In Variabili di sistema, seleziona Percorso, quindi scegli Modifica. Nella finestra Modifica variabile di ambiente, puoi visualizzare ogni percorso su una riga separata.
   f. Controlla se è presente il percorso della `bin` cartella di installazione di Java. Il percorso potrebbe essere simile all'esempio seguente.

```
C:\\Program Files\\Amazon Corretto\\jdk11.0.13_8\\bin
```

g. Se la `bin` cartella di installazione Java non è presente in Path, scegliete Nuovo per aggiungerla, quindi scegliete OK.

3. Aprite il prompt dei comandi di Windows (`cmd.exe`) come amministratore.
4. Crea l'utente predefinito nell' LocalSystem account sul dispositivo Windows. Sostituisci *la password* con una password sicura.

```
net user /add ggc_user password
```

> **Tip**
>
> A seconda della configurazione di Windows, la password dell'utente potrebbe essere impostata per scadere in date future. Per garantire che le tue applicazioni Greengrass continuino a funzionare, tieni traccia della scadenza della password e aggiornala prima che scada. Puoi anche impostare la password dell'utente in modo che non scada mai.
>
> - Per verificare la scadenza di un utente e della relativa password, esegui il comando seguente.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Per impostare la password di un utente in modo che non scada mai, esegui il comando seguente.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```
>
> - Se utilizzi Windows 10 o versioni successive in cui il wmiccomando è obsoleto, esegui il comando seguente. PowerShell
>
>   ```
>   Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
>    'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
>   ```

5. Scarica e installa l'PsExecutilità di Microsoft sul dispositivo.
6. Utilizzate l' PsExec utilità per memorizzare il nome utente e la password per l'utente predefinito nell'istanza di Credential Manager per l' LocalSystem account. Sostituisci la *password* con la password dell'utente impostata in precedenza.

```
psexec -s cmd /c cmdkey /generic:ggc_user /user:ggc_user /pass:password
```

Se si PsExec License Agreementapre, scegli Acceptdi accettare la licenza ed esegui il comando.

> **Note**
>
> Sui dispositivi Windows, l' LocalSystem account esegue il Greengrass nucleus ed è necessario utilizzare l' PsExec utilità per memorizzare le informazioni utente predefinite nell'account. LocalSystem L'utilizzo dell'applicazione Credential Manager archivia queste informazioni nell'account Windows dell'utente attualmente connesso, anziché nell'account. LocalSystem

Scaricate il software Core AWS IoT Greengrass

È possibile scaricare la versione più recente del software AWS IoT Greengrass Core dal seguente indirizzo:

- https://d2s8p88vqu9w66.cloudfront.net/releases/ greengrass-nucleus-latest .zip

> **Note**
>
> È possibile scaricare una versione specifica del software AWS IoT Greengrass Core dal seguente percorso. Sostituisci la *versione* con la versione da scaricare.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Per scaricare il software AWS IoT Greengrass Core

1. Sul dispositivo principale, scaricate il software AWS IoT Greengrass Core in un file denominato`greengrass-nucleus-latest.zip`.

Linux or Unix

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
latest.zip > greengrass-nucleus-latest.zip
```

Windows Command Prompt (CMD)

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
latest.zip > greengrass-nucleus-latest.zip
```

PowerShell

```
iwr -Uri https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
latest.zip -OutFile greengrass-nucleus-latest.zip
```

Scaricando questo software accetti l'Accordo di licenza del software Greengrass Core.

2. (Facoltativo) Per verificare la firma del software Greengrass nucleus

> **Note**
>
> Questa funzionalità è disponibile con Greengrass nucleus versione 2.9.5 e successive.

a. Usa il seguente comando per verificare la firma del tuo artefatto Greengrass nucleus:

Linux or Unix

```
jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
```

Windows Command Prompt (CMD)

Il nome del file potrebbe avere un aspetto diverso a seconda della versione di JDK installata. *jdk17.0.6_10*Sostituiscilo con la versione JDK che hai installato.

```
"C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
verify -certs -verbose greengrass-nucleus-latest.zip
```

PowerShell

Il nome del file potrebbe avere un aspetto diverso a seconda della versione di JDK installata. *jdk17.0.6_10*Sostituiscilo con la versione JDK che hai installato.

```
'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
verify -certs -verbose greengrass-nucleus-latest.zip
```

b. L'`jarsigner`invocazione produce un output che indica i risultati della verifica.

i. Se il file zip Greengrass nucleus è firmato, l'output contiene la seguente dichiarazione:

```
jar verified.
```

ii. Se il file zip Greengrass nucleus non è firmato, l'output contiene la seguente dichiarazione:

```
jar is unsigned.
```

c. Se hai fornito l'`-certs`opzione Jarsigner insieme alle `-verbose` opzioni `-verify` e, l'output include anche informazioni dettagliate sul certificato del firmatario.

3. Decomprimi il software AWS IoT Greengrass Core in una cartella sul tuo dispositivo. Sostituiscilo *GreengrassInstaller*con la cartella che desideri utilizzare.

Linux or Unix

```
unzip greengrass-nucleus-latest.zip -d GreengrassInstaller && rm greengrass-
nucleus-latest.zip
```

Windows Command Prompt (CMD)

```
mkdir GreengrassInstaller && tar -xf greengrass-nucleus-latest.zip -
C GreengrassInstaller && del greengrass-nucleus-latest.zip
```

PowerShell

```
Expand-Archive -Path greengrass-nucleus-latest.zip -DestinationPath .\
\GreengrassInstaller
rm greengrass-nucleus-latest.zip
```

4. (Facoltativo) Eseguite il comando seguente per visualizzare la versione del software AWS IoT Greengrass Core.

```
java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
```

> **Important**
>
> Se installi una versione del nucleo Greengrass precedente alla v2.4.0, non rimuovere questa cartella dopo aver installato il software Core. AWS IoT Greengrass Il software AWS IoT Greengrass Core utilizza i file in questa cartella per l'esecuzione.
> Se hai scaricato la versione più recente del software, installi la versione 2.4.0 o successiva e puoi rimuovere questa cartella dopo aver installato il software AWS IoT Greengrass Core.

Scarica il plug-in per il provisioning AWS IoT della flotta

Puoi scaricare l'ultima versione del plug-in per il provisioning AWS IoT della flotta dal seguente indirizzo:

- https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass - /fleetprovisioningbyclaim-latest.jar FleetProvisioningByClaim

> **Note**
>
> È possibile scaricare una versione specifica del plug-in per il provisioning AWS IoT della flotta dalla seguente posizione. Sostituisci la *versione* con la versione da scaricare. Per ulteriori informazioni su ciascuna versione del plug-in Fleet Provisioning, consultaAWS IoT log delle modifiche del plugin per il provisioning della flotta.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass-
> FleetProvisioningByClaim/fleetprovisioningbyclaim-version.jar
> ```

Il plug-in per il provisioning della flotta è open source. Per visualizzarne il codice sorgente, consulta il plug-in AWS IoT Fleet Provisioning su. GitHub

Per scaricare il plug-in per il provisioning AWS IoT della flotta

- Sul tuo dispositivo, scarica il plug-in AWS IoT Fleet Provisioning in un file denominato. `aws.greengrass.FleetProvisioningByClaim.jar` Sostituiscilo *`GreengrassInstaller`*con la cartella che desideri utilizzare.

  Linux or Unix

  ```
  curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass-
  FleetProvisioningByClaim/fleetprovisioningbyclaim-latest.jar
   > GreengrassInstaller/aws.greengrass.FleetProvisioningByClaim.jar
  ```

  Windows Command Prompt (CMD)

  ```
  curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass-
  FleetProvisioningByClaim/fleetprovisioningbyclaim-latest.jar
   > GreengrassInstaller/aws.greengrass.FleetProvisioningByClaim.jar
  ```

  PowerShell

  ```
  iwr -Uri https://d2s8p88vqu9w66.cloudfront.net/releases/aws-greengrass-
  FleetProvisioningByClaim/fleetprovisioningbyclaim-latest.jar -
  OutFile GreengrassInstaller/aws.greengrass.FleetProvisioningByClaim.jar
  ```

  Scaricando questo software accetti l'Accordo di licenza del software Greengrass Core.

Installa il software AWS IoT Greengrass Core

Esegui il programma di installazione con argomenti che specificano le seguenti azioni:

- Esegui l'installazione da un file di configurazione parziale che specifica di utilizzare il plug-in fleet provisioning per il provisioning delle risorse. AWS Il software AWS IoT Greengrass Core utilizza un file di configurazione che specifica la configurazione di ogni componente Greengrass sul dispositivo. Il programma di installazione crea un file di configurazione completo a partire dal file di configurazione parziale fornito dall'utente e dalle AWS risorse create dal plug-in Fleet Provisioning.
- Specificare di utilizzare l'utente `ggc_user` del sistema per eseguire i componenti software sul dispositivo principale. Sui dispositivi Linux, questo comando specifica anche di utilizzare il gruppo

di `ggc_group` sistema e il programma di installazione crea automaticamente l'utente e il gruppo di sistema.

- Configura il software AWS IoT Greengrass Core come servizio di sistema che viene eseguito all'avvio. Sui dispositivi Linux, ciò richiede il sistema di inizializzazione Systemd.

> **Important**
>
> Sui dispositivi Windows core, è necessario configurare il software AWS IoT Greengrass Core come servizio di sistema.

Per ulteriori informazioni sugli argomenti che è possibile specificare, vedereArgomenti dell'installatore.

> **Note**
>
> Se utilizzi AWS IoT Greengrass un dispositivo con memoria limitata, puoi controllare la quantità di memoria utilizzata dal software AWS IoT Greengrass Core. Per controllare l'allocazione della memoria, è possibile impostare le opzioni relative alla dimensione dell'heap JVM nel parametro di `jvmOptions` configurazione del componente nucleus. Per ulteriori informazioni, consulta Controlla l'allocazione della memoria con le opzioni JVM.

Per installare il software Core AWS IoT Greengrass

1. Controlla la versione del software AWS IoT Greengrass Core.

   - Sostituisci *`GreengrassInstaller`*con il percorso della cartella che contiene il software.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

2. Utilizzate un editor di testo per creare un file di configurazione denominato `config.yaml` da fornire all'installatore.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

   ```
   nano GreengrassInstaller/config.yaml
   ```

Copiate il seguente contenuto YAML nel file. Questo file di configurazione parziale specifica i parametri per il plug-in Fleet Provisioning. Per ulteriori informazioni sulle opzioni che è possibile specificare, vedere. Configurare il plug-in per il provisioning AWS IoT della flotta

Linux or Unix

```
---
services:
  aws.greengrass.Nucleus:
    version: "2.12.3"
  aws.greengrass.FleetProvisioningByClaim:
    configuration:
      rootPath: "/greengrass/v2"
      awsRegion: "us-west-2"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
      iotCredentialEndpoint: "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      provisioningTemplate: "GreengrassFleetProvisioningTemplate"
      claimCertificatePath: "/greengrass/v2/claim-certs/claim.pem.crt"
      claimCertificatePrivateKeyPath: "/greengrass/v2/claim-certs/
claim.private.pem.key"
      rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
      templateParameters:
        ThingName: "MyGreengrassCore"
        ThingGroupName: "MyGreengrassCoreGroup"
```

Windows

```
---
services:
  aws.greengrass.Nucleus:
    version: "2.12.3"
  aws.greengrass.FleetProvisioningByClaim:
    configuration:
      rootPath: "C:\\greengrass\\v2"
      awsRegion: "us-west-2"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
      iotCredentialEndpoint: "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      provisioningTemplate: "GreengrassFleetProvisioningTemplate"
```

```
      claimCertificatePath: "C:\\greengrass\\v2\\claim-certs\\claim.pem.crt"
      claimCertificatePrivateKeyPath: "C:\\greengrass\\v2\\claim-certs\
\claim.private.pem.key"
      rootCaPath: "C:\\greengrass\\v2\\AmazonRootCA1.pem"
      templateParameters:
        ThingName: "MyGreengrassCore"
        ThingGroupName: "MyGreengrassCoreGroup"
```

Successivamente, esegui queste operazioni:

- Sostituire la versione *2.12.3* con la versione del software AWS IoT Greengrass Core.
- Sostituisci ogni istanza di */greengrass/v2* o *C:\greengrass\v2* con la cartella principale di Greengrass.

> **Note**
>
> Sui dispositivi Windows, è necessario specificare i separatori di percorso come doppie barre rovesciate (\\), ad esempio. `C:\\greengrass\\v2`

- Sostituisci *us-west-2* con AWS la regione in cui hai creato il modello di provisioning e altre risorse.
- Sostituiscilo `iotDataEndpoint` con il tuo endpoint di dati. AWS IoT
- Sostituisci l'`iotCredentialEndpoint`endpoint con AWS IoT le tue credenziali.
- Sostituisci *GreengrassCoreTokenExchangeRoleAlias*con il nome dell'alias del ruolo di scambio di token.
- Sostituiscilo *GreengrassFleetProvisioningTemplate*con il nome del modello di approvvigionamento della flotta.
- Sostituisci il `claimCertificatePath` con il percorso del certificato di richiesta sul dispositivo.
- Sostituiscilo `claimCertificatePrivateKeyPath` con il percorso della chiave privata del certificato di richiesta sul dispositivo.
- Sostituisci i parametri del modello (`templateParameters`) con i valori da utilizzare per il provisioning del dispositivo. Questo esempio si riferisce al modello di esempio che definisce `ThingName` i `ThingGroupName` parametri.

Note

In questo file di configurazione, è possibile personalizzare altre opzioni di configurazione come le porte e il proxy di rete da utilizzare, come illustrato nell'esempio seguente. Per ulteriori informazioni, vedere Greengrass nucleus configuration.

Linux or Unix

```
---
services:
  aws.greengrass.Nucleus:
    version: "2.12.3"
    configuration:
      mqtt:
        port: 443
      greengrassDataPlanePort: 443
      networkProxy:
        noProxyAddresses: "http://192.168.0.1,www.example.com"
        proxy:
          url: "http://my-proxy-server:1100"
          username: "Mary_Major"
          password: "pass@word1357"
  aws.greengrass.FleetProvisioningByClaim:
    configuration:
      rootPath: "/greengrass/v2"
      awsRegion: "us-west-2"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-
west-2.amazonaws.com"
      iotCredentialEndpoint: "device-credentials-
prefix.credentials.iot.us-west-2.amazonaws.com"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      provisioningTemplate: "GreengrassFleetProvisioningTemplate"
      claimCertificatePath: "/greengrass/v2/claim-certs/claim.pem.crt"
      claimCertificatePrivateKeyPath: "/greengrass/v2/claim-certs/
claim.private.pem.key"
      rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
      templateParameters:
        ThingName: "MyGreengrassCore"
        ThingGroupName: "MyGreengrassCoreGroup"
      mqttPort: 443
      proxyUrl: "http://my-proxy-server:1100"
```

```
      proxyUserName: "Mary_Major"
      proxyPassword: "pass@word1357"
```

Windows

```
---
services:
  aws.greengrass.Nucleus:
    version: "2.12.3"
    configuration:
      mqtt:
        port: 443
      greengrassDataPlanePort: 443
      networkProxy:
        noProxyAddresses: "http://192.168.0.1,www.example.com"
        proxy:
          url: "http://my-proxy-server:1100"
          username: "Mary_Major"
          password: "pass@word1357"
  aws.greengrass.FleetProvisioningByClaim:
    configuration:
      rootPath: "C:\\greengrass\\v2"
      awsRegion: "us-west-2"
      iotDataEndpoint: "device-data-prefix-ats.iot.us-
west-2.amazonaws.com"
      iotCredentialEndpoint: "device-credentials-
prefix.credentials.iot.us-west-2.amazonaws.com"
      iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
      provisioningTemplate: "GreengrassFleetProvisioningTemplate"
      claimCertificatePath: "C:\\greengrass\\v2\\claim-certs\
\claim.pem.crt"
      claimCertificatePrivateKeyPath: "C:\\greengrass\\v2\\claim-certs\
\claim.private.pem.key"
      rootCaPath: "C:\\greengrass\\v2\\AmazonRootCA1.pem"
      templateParameters:
        ThingName: "MyGreengrassCore"
        ThingGroupName: "MyGreengrassCoreGroup"
      mqttPort: 443
      proxyUrl: "http://my-proxy-server:1100"
      proxyUserName: "Mary_Major"
      proxyPassword: "pass@word1357"
```

Per utilizzare un proxy HTTPS, è necessario utilizzare la versione 1.1.0 o successiva del plug-in Fleet Provisioning. È inoltre necessario specificare quanto segue`system`, come `rootCaPath` illustrato nell'esempio seguente.

Linux or Unix

```
---
system:
  rootCaPath: "/greengrass/v2/AmazonRootCA1.pem"
services:
  ...
```

Windows

```
---
system:
  rootCaPath: "C:\\greengrass\\v2\\AmazonRootCA1.pem"
services:
  ...
```

3. Eseguire il programma di installazione. Specificate `--trusted-plugin` di fornire il plug-in Fleet Provisioning e specificate `--init-config` di fornire il file di configurazione.

   - Sostituisci */greengrass/v2* con la cartella principale di Greengrass.
   - Sostituisci ogni istanza di *GreengrassInstaller*con la cartella in cui hai decompresso il programma di installazione.

   Linux or Unix

   ```
   sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
     -jar ./GreengrassInstaller/lib/Greengrass.jar \
     --trusted-plugin ./GreengrassInstaller/
   aws.greengrass.FleetProvisioningByClaim.jar \
     --init-config ./GreengrassInstaller/config.yaml \
     --component-default-user ggc_user:ggc_group \
     --setup-system-service true
   ```

Windows Command Prompt (CMD)

```
java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" ^
  -jar ./GreengrassInstaller/lib/Greengrass.jar ^
  --trusted-plugin ./GreengrassInstaller/
aws.greengrass.FleetProvisioningByClaim.jar ^
  --init-config ./GreengrassInstaller/config.yaml ^
  --component-default-user ggc_user ^
  --setup-system-service true
```

PowerShell

```
java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" `
  -jar ./GreengrassInstaller/lib/Greengrass.jar `
  --trusted-plugin ./GreengrassInstaller/
aws.greengrass.FleetProvisioningByClaim.jar `
  --init-config ./GreengrassInstaller/config.yaml `
  --component-default-user ggc_user `
  --setup-system-service true
```

Important

Nei dispositivi Windows Core, è necessario specificare `--setup-system-service true` di configurare il software AWS IoT Greengrass Core come servizio di sistema.

Se si specifica`--setup-system-service true`, il programma di installazione stampa `Successfully set up Nucleus as a system service` se ha configurato ed eseguito il software come servizio di sistema. Altrimenti, il programma di installazione non emette alcun messaggio se installa il software correttamente.

Note

Non è possibile utilizzare l'`deploy-dev-tools`argomento per distribuire strumenti di sviluppo locali quando si esegue il programma di installazione senza l'argomento. `--provision true` Per informazioni sulla distribuzione della CLI Greengrass direttamente sul dispositivo, consulta. Interfaccia a riga di comando Greengrass

4. Verifica l'installazione visualizzando i file nella cartella principale.

   Linux or Unix

   ```
   ls /greengrass/v2
   ```

   Windows Command Prompt (CMD)

   ```
   dir C:\greengrass\v2
   ```

   PowerShell

   ```
   ls C:\greengrass\v2
   ```

   Se l'installazione è riuscita, la cartella principale contiene diverse cartelle, ad esempio `configpackages`, `elogs`.

Se avete installato il software AWS IoT Greengrass Core come servizio di sistema, il programma di installazione esegue il software automaticamente. In caso contrario, è necessario eseguire il software manualmente. Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core.

Per ulteriori informazioni su come configurare e utilizzare il software e AWS IoT Greengrass, vedere quanto segue:

- Configurare il software AWS IoT Greengrass Core
- Sviluppa AWS IoT Greengrass componenti
- Implementazione AWS IoT Greengrass dei componenti sui dispositivi
- Interfaccia a riga di comando Greengrass

Configura il provisioning AWS IoT della flotta per i dispositivi core Greengrass

Per installare il software AWS IoT Greengrass Core con il provisioning della flotta, devi prima configurare le seguenti risorse nel tuo. Account AWS Queste risorse consentono ai dispositivi di registrarsi AWS IoT e funzionare come dispositivi core Greengrass. Segui una volta i passaggi in questa sezione per creare e configurare queste risorse nel tuoAccount AWS.

- Un ruolo IAM per lo scambio di token, utilizzato dai dispositivi principali per autorizzare le chiamate ai AWS servizi.
- Un alias di AWS IoT ruolo che rimanda al ruolo di scambio di token.
- (Facoltativo) Una AWS IoT policy utilizzata dai dispositivi principali per autorizzare le chiamate ai servizi AWS IoT andAWS IoT Greengrass. Questa AWS IoT politica deve consentire l'`iot:AssumeRoleWithCertificate`autorizzazione per l'alias del AWS IoT ruolo che punta al ruolo di scambio di token.

  È possibile utilizzare un'unica AWS IoT policy per tutti i dispositivi principali del parco dispositivi oppure configurare il modello di provisioning del parco veicoli per creare una AWS IoT policy per ogni dispositivo principale.
- Un modello di approvvigionamento del AWS IoT parco veicoli. Questo modello deve specificare quanto segue:
  - Qualsiasi AWS IoT cosa, risorsa. È possibile specificare un elenco di gruppi di oggetti esistenti per distribuire i componenti su ciascun dispositivo quando è online.
  - Una risorsa AWS IoT politica. Questa risorsa può definire una delle seguenti proprietà:
    - Il nome di una AWS IoT politica esistente. Se scegli questa opzione, i dispositivi principali che crei con questo modello utilizzano la stessa AWS IoT politica e puoi gestirne le autorizzazioni come flotta.
    - Un documento AWS IoT di policy. Se scegli questa opzione, ogni dispositivo principale creato a partire da questo modello utilizza una AWS IoT policy unica e puoi gestire le autorizzazioni per ogni singolo dispositivo principale.
  - Una risorsa AWS IoT certificata. Questa risorsa di certificato deve utilizzare il `AWS::IoT::Certificate::Id` parametro per allegare il certificato al dispositivo principale. Per ulteriori informazioni, consulta J ust-in-time provisioning nella AWS IoTDeveloper Guide.
- Un certificato di richiesta di AWS IoT approvvigionamento e una chiave privata per il modello di approvvigionamento della flotta. Puoi incorporare questo certificato e la chiave privata nei dispositivi durante la produzione, in modo che i dispositivi possano registrarsi e rifornirsi autonomamente quando sono online.

> **Important**
>
> Il provisioning dichiara che le chiavi private devono essere protette in ogni momento, anche sui dispositivi core Greengrass. Ti consigliamo di utilizzare i CloudWatch parametri e i log di Amazon per monitorare eventuali indicazioni di uso improprio, come l'uso non

autorizzato del certificato di attestazione per il provisioning dei dispositivi. Se rilevi un uso improprio, disattiva il certificato di richiesta di approvvigionamento in modo che non possa essere utilizzato per il provisioning dei dispositivi. Per ulteriori informazioni, consulta Monitoring AWS IoT nella Developer Guide. AWS IoT Core
Per aiutarti a gestire meglio il numero di dispositivi e i dispositivi che si registrano automaticamente nel tuo sistemaAccount AWS, puoi specificare un hook di pre-provisioning quando crei un modello di provisioning del parco veicoli. Un hook di pre-provisioning è una AWS Lambda funzione che convalida i parametri del modello forniti dai dispositivi durante la registrazione. Ad esempio, è possibile creare un hook di pre-provisioning che controlli l'ID di un dispositivo confrontandolo con un database per verificare che il dispositivo disponga dell'autorizzazione al provisioning. Per ulteriori informazioni, consulta Pre-provisioning hook nella Developer Guide. AWS IoT Core

- Una AWS IoT politica da allegare al certificato di richiesta di approvvigionamento per consentire ai dispositivi di registrarsi e utilizzare il modello di provisioning del parco veicoli.

Argomenti

- Crea un ruolo di scambio di token
- Creazione di una policy AWS IoT
- Crea un modello di approvvigionamento del parco veicoli
- Crea un certificato di richiesta di approvvigionamento e una chiave privata

Crea un ruolo di scambio di token

I dispositivi core Greengrass utilizzano un ruolo di servizio IAM, chiamato token exchange role, per autorizzare le chiamate ai servizi. AWS Il dispositivo utilizza il provider di AWS IoT credenziali per ottenere AWS credenziali temporanee per questo ruolo, che consente al dispositivo di interagireAWS IoT, inviare log ad Amazon CloudWatch Logs e scaricare elementi dei componenti personalizzati da Amazon S3. Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

Si utilizza un alias di AWS IoT ruolo per configurare il ruolo di scambio di token per i dispositivi principali Greengrass. Gli alias di ruolo consentono di modificare il ruolo di scambio di token per un dispositivo ma mantengono invariata la configurazione del dispositivo. Per ulteriori informazioni, consulta Autorizzazione delle chiamate dirette ai AWS servizi nella Guida per gli AWS IoT Coresviluppatori.

In questa sezione, crei un ruolo IAM per lo scambio di token e un alias di AWS IoT ruolo che rimanda al ruolo. Se hai già configurato un dispositivo principale Greengrass, puoi utilizzare il ruolo di scambio di token e l'alias del ruolo invece di crearne di nuovi.

Per creare un ruolo IAM per lo scambio di token

1. Crea un ruolo IAM che il tuo dispositivo possa utilizzare come ruolo di scambio di token. Esegui questa operazione:

   a. Crea un file che contenga il documento sulla politica di fiducia richiesto dal ruolo di scambio di token.

      Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

      ```
      nano device-role-trust-policy.json
      ```

      Copiate il seguente codice JSON nel file.

      ```
      {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "credentials.iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      }
      ```

   b. Crea il ruolo di scambio di token con il documento sulla politica di fiducia.

      - Sostituisci *greengrassV2 TokenExchangeRole* con il nome del ruolo IAM da creare.

      ```
      aws iam create-role --role-name GreengrassV2TokenExchangeRole --assume-role-
      policy-document file://device-role-trust-policy.json
      ```

      La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "Role": {
    "Path": "/",
    "RoleName": "GreengrassV2TokenExchangeRole",
    "RoleId": "AROAZ2YMUHYHK5OKM77FB",
    "Arn": "arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole",
    "CreateDate": "2021-02-06T00:13:29+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "credentials.iot.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
```

c. Crea un file che contenga il documento sulla politica di accesso richiesto dal ruolo di scambio di token.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

   ```
   nano device-role-access-policy.json
   ```

   Copiate il seguente codice JSON nel file.

   ```
   {
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Action": [
           "logs:CreateLogGroup",
           "logs:CreateLogStream",
           "logs:PutLogEvents",
           "logs:DescribeLogStreams",
   ```

```
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Questa politica di accesso non consente l'accesso agli artefatti dei componenti nei bucket S3. Per distribuire componenti personalizzati che definiscono gli artefatti in Amazon S3, devi aggiungere autorizzazioni al ruolo per consentire al dispositivo principale di recuperare gli artefatti dei componenti. Per ulteriori informazioni, consulta Consenti l'accesso ai bucket S3 per gli artefatti dei componenti.
> Se non disponi ancora di un bucket S3 per gli artefatti dei componenti, puoi aggiungere queste autorizzazioni in un secondo momento dopo aver creato un bucket.

d. Crea la policy IAM dal documento di policy.

- Sostituisci *GreenGrassV2 TokenExchangeRoleAccess* con il nome della policy IAM da creare.

```
aws iam create-policy --policy-name GreengrassV2TokenExchangeRoleAccess --
policy-document file://device-role-access-policy.json
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "Policy": {
    "PolicyName": "GreengrassV2TokenExchangeRoleAccess",
    "PolicyId": "ANPAZ2YMUHYHACI7C5Z66",
    "Arn": "arn:aws:iam::123456789012:policy/
GreengrassV2TokenExchangeRoleAccess",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
```

```
    "CreateDate": "2021-02-06T00:37:17+00:00",
    "UpdateDate": "2021-02-06T00:37:17+00:00"
  }
}
```

e. Allega la policy IAM al ruolo di scambio di token.

- Sostituisci *greengrassV2 TokenExchangeRole* con il nome del ruolo IAM.
- Sostituisci l'ARN della policy con l'ARN della policy IAM che hai creato nel passaggio precedente.

```
aws iam attach-role-policy --role-name GreengrassV2TokenExchangeRole --policy-
arn arn:aws:iam::123456789012:policy/GreengrassV2TokenExchangeRoleAccess
```

Il comando non ha alcun output se la richiesta ha esito positivo.

2. Crea un alias di AWS IoT ruolo che punti al ruolo di scambio di token.

- Sostituiscilo *GreengrassCoreTokenExchangeRoleAlias*con il nome dell'alias del ruolo da creare.
- Sostituisci il ruolo ARN con l'ARN del ruolo IAM creato nel passaggio precedente.

```
aws iot create-role-alias --role-alias GreengrassCoreTokenExchangeRoleAlias --role-
arn arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "roleAlias": "GreengrassCoreTokenExchangeRoleAlias",
  "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
}
```

> **Note**
>
> Per creare un alias di ruolo, devi disporre dell'autorizzazione a passare il ruolo IAM per lo scambio di token a. AWS IoT Se ricevi un messaggio di errore quando tenti di creare un alias di ruolo, verifica che AWS l'utente disponga di questa autorizzazione.

Per ulteriori informazioni, consulta Concessione a un utente delle autorizzazioni per il trasferimento di un ruolo a un AWS servizio nella Guida per l'AWS Identity and Access Managementutente.

Creazione di una policy AWS IoT

Dopo aver registrato un dispositivo come AWS IoT oggetto, quel dispositivo può utilizzare un certificato digitale con cui autenticarsi. AWS Questo certificato include una o più AWS IoT politiche che definiscono le autorizzazioni che un dispositivo può utilizzare con il certificato. Queste politiche consentono al dispositivo di comunicare con AWS IoT eAWS IoT Greengrass.

Con AWS IoT Fleet Provisioning, i dispositivi si connettono AWS IoT per creare e scaricare un certificato del dispositivo. Nel modello di provisioning del parco veicoli creato nella sezione successiva, puoi specificare se AWS IoT allega la stessa AWS IoT policy ai certificati di tutti i dispositivi o crea una nuova policy per ogni dispositivo.

In questa sezione, crei una AWS IoT policy da AWS IoT allegare ai certificati di tutti i dispositivi. Con questo approccio, puoi gestire le autorizzazioni per tutti i dispositivi come una flotta. Se preferisci creare una nuova AWS IoT politica per ogni dispositivo, puoi saltare questa sezione e fare riferimento alla politica in essa contenuta quando definisci il modello del tuo parco veicoli.

Per creare una policy AWS IoT

- Crea una AWS IoT politica che definisca le AWS IoT autorizzazioni per la tua flotta di dispositivi principali Greengrass. La seguente politica consente l'accesso a tutti gli argomenti MQTT e alle operazioni di Greengrass, in modo che il dispositivo funzioni con applicazioni personalizzate e modifiche future che richiedono nuove operazioni Greengrass. Questa politica consente anche l'`iot:AssumeRoleWithCertificate`autorizzazione, che consente ai dispositivi di utilizzare il ruolo di scambio di token creato nella sezione precedente. Puoi limitare questa politica in base al tuo caso d'uso. Per ulteriori informazioni, consulta AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali.

  Esegui questa operazione:

  a. Crea un file che contenga il documento di AWS IoT policy richiesto dai dispositivi core Greengrass.

Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

```
nano greengrass-v2-iot-policy.json
```

Copiate il seguente codice JSON nel file.

- Sostituisci la `iot:AssumeRoleWithCertificate` risorsa con l'ARN dell'alias del AWS IoT ruolo creato nella sezione precedente.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "iot:Connect",
        "greengrass:*"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:AssumeRoleWithCertificate",
      "Resource": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
    }
  ]
}
```

b. Crea una AWS IoT politica dal documento di policy.

- Sostituisci *GreengrassV2IoT ThingPolicy* con il nome della politica da creare.

```
aws iot create-policy --policy-name GreengrassV2IoTThingPolicy --policy-
document file://greengrass-v2-iot-policy.json
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "policyName": "GreengrassV2IoTThingPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
  "policyDocument": "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
      {
        \"Effect\": \"Allow\",
        \"Action\": [
          \"iot:Publish\",
          \"iot:Subscribe\",
          \"iot:Receive\",
          \"iot:Connect\",
          \"greengrass:*\"
        ],
        \"Resource\": [
          \"*\"
        ]
      },
      {
        \"Effect\": \"Allow\",
        \"Action\": \"iot:AssumeRoleWithCertificate\",
        \"Resource\": \"arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias\"
      }
    ]
  }",
  "policyVersionId": "1"
}
```

Crea un modello di approvvigionamento del parco veicoli

AWS IoTi modelli di provisioning della flotta definiscono come fornire AWS IoT oggetti, politiche e certificati. Per effettuare il provisioning dei dispositivi core Greengrass con il plug-in fleet provisioning, è necessario creare un modello che specifichi quanto segue:

- Una cosa, una risorsaAWS IoT. È possibile specificare un elenco di gruppi di oggetti esistenti per distribuire i componenti su ciascun dispositivo quando è online.
- Una risorsa AWS IoT politica. Questa risorsa può definire una delle seguenti proprietà:
  - Il nome di una AWS IoT politica esistente. Se scegli questa opzione, i dispositivi principali che crei con questo modello utilizzano la stessa AWS IoT politica e puoi gestirne le autorizzazioni come flotta.
  - Un documento AWS IoT di policy. Se scegli questa opzione, ogni dispositivo principale creato a partire da questo modello utilizza una AWS IoT policy unica e puoi gestire le autorizzazioni per ogni singolo dispositivo principale.
- Una risorsa AWS IoT certificata. Questa risorsa di certificato deve utilizzare il `AWS::IoT::Certificate::Id` parametro per allegare il certificato al dispositivo principale. Per ulteriori informazioni, consulta J ust-in-time provisioning nella AWS IoTDeveloper Guide.

Nel modello, è possibile specificare di aggiungere l'AWS IoToggetto a un elenco di gruppi di oggetti esistenti. Quando il dispositivo principale si connette AWS IoT Greengrass per la prima volta, riceve le distribuzioni Greengrass per ogni gruppo di cose di cui è membro. Puoi utilizzare i gruppi di oggetti per distribuire il software più recente su ciascun dispositivo non appena è online. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.

Il AWS IoT servizio richiede le autorizzazioni per creare e aggiornare AWS IoT le risorse nei dispositivi Account AWS durante il provisioning. Per consentire l'accesso al AWS IoT servizio, crei un ruolo IAM e lo fornisci quando crei il modello. AWS IoTfornisce una policy gestita che consente l'accesso a tutte le autorizzazioni che AWS IoT potrebbero essere utilizzate per il provisioning dei dispositivi. AWSIoTThingsRegistration È possibile utilizzare questa policy gestita o creare una policy personalizzata che definisca le autorizzazioni nella policy gestita in base al proprio caso d'uso.

In questa sezione, crei un ruolo IAM che consente di AWS IoT effettuare il provisioning di risorse per i dispositivi e crei un modello di provisioning della flotta che utilizza quel ruolo IAM.

Per creare un modello di provisioning della flotta

1. Crea un ruolo IAM che AWS IoT possa pretendere di fornire risorse nel tuoAccount AWS. Esegui questa operazione:

   a. Crea un file che contenga il documento sulla politica di fiducia che AWS IoT consente di assumere il ruolo.

      Ad esempio, su un sistema basato su Linux, è possibile eseguire il seguente comando per utilizzare GNU nano per creare il file.

      ```
      nano aws-iot-trust-policy.json
      ```

      Copiate il seguente codice JSON nel file.

      ```
      {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      }
      ```

   b. Crea un ruolo IAM con il documento sulla politica di fiducia.

      - Sostituiscilo *GreengrassFleetProvisioningRole*con il nome del ruolo IAM da creare.

      ```
      aws iam create-role --role-name GreengrassFleetProvisioningRole --assume-role-
      policy-document file://aws-iot-trust-policy.json
      ```

      La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

      ```
      {
        "Role": {
      ```

```
    "Path": "/",
    "RoleName": "GreengrassFleetProvisioningRole",
    "RoleId": "AROAZ2YMUHYHK5OKM77FB",
    "Arn": "arn:aws:iam::123456789012:role/GreengrassFleetProvisioningRole",
    "CreateDate": "2021-07-26T00:15:12+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "iot.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
}
```

c. Esamina la AWSIoTThingsRegistrationpolitica, che consente l'accesso a tutte le autorizzazioni che AWS IoT potrebbero essere utilizzate per il provisioning dei dispositivi. Puoi utilizzare questa policy gestita o creare una policy personalizzata che definisca autorizzazioni limitate per il tuo caso d'uso. Se scegli di creare una politica personalizzata, fallo ora.

d. Allega la policy IAM al ruolo di provisioning della flotta.

   - Sostituisci *GreengrassFleetProvisioningRole* con il nome del ruolo IAM.
   - Se hai creato una policy personalizzata nel passaggio precedente, sostituisci l'ARN della policy con l'ARN della policy IAM da utilizzare.

```
aws iam attach-role-policy --role-name GreengrassFleetProvisioningRole --
policy-arn arn:aws:iam::aws:policy/service-role/AWSIoTThingsRegistration
```

   Il comando non ha alcun output se la richiesta ha esito positivo.

2. (Facoltativo) Crea un hook di pre-provisioning, che è una AWS Lambda funzione che convalida i parametri del modello forniti dai dispositivi durante la registrazione. Puoi utilizzare un hook di pre-provisioning per avere un maggiore controllo su quali e quanti dispositivi sono integrati nel

tuo dispositivo. Account AWS Per ulteriori informazioni, consulta gli hook di pre-provisioning nella Developer Guide. AWS IoT Core

3. Crea un modello di provisioning del parco veicoli. Esegui questa operazione:

   a. Crea un file per contenere il documento del modello di approvvigionamento.

      Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

      ```
      nano greengrass-fleet-provisioning-template.json
      ```

      Scrivete il documento modello di provisioning. È possibile iniziare dal seguente esempio di modello di provisioning, che specifica di creare un AWS IoT oggetto con le seguenti proprietà:

      - Il nome dell'oggetto è il valore specificato nel parametro `ThingName` template.
      - L'oggetto è un membro del gruppo di oggetti specificato nel parametro `ThingGroupName` template. Il gruppo di cose deve esistere nel tuoAccount AWS.
      - Al certificato dell'oggetto è `GreengrassV2IoTThingPolicy` allegata la AWS IoT politica denominata.

      Per ulteriori informazioni, consulta Provisioning templates nella AWS IoT CoreDeveloper Guide.

      ```
      {
        "Parameters": {
          "ThingName": {
            "Type": "String"
          },
          "ThingGroupName": {
            "Type": "String"
          },
          "AWS::IoT::Certificate::Id": {
            "Type": "String"
          }
        },
        "Resources": {
          "MyThing": {
            "OverrideSettings": {
      ```

```
        "AttributePayload": "REPLACE",
        "ThingGroups": "REPLACE",
        "ThingTypeName": "REPLACE"
      },
      "Properties": {
        "AttributePayload": {},
        "ThingGroups": [
          {
            "Ref": "ThingGroupName"
          }
        ],
        "ThingName": {
          "Ref": "ThingName"
        }
      },
      "Type": "AWS::IoT::Thing"
    },
    "MyPolicy": {
      "Properties": {
        "PolicyName": "GreengrassV2IoTThingPolicy"
      },
      "Type": "AWS::IoT::Policy"
    },
    "MyCertificate": {
      "Properties": {
        "CertificateId": {
          "Ref": "AWS::IoT::Certificate::Id"
        },
        "Status": "Active"
      },
      "Type": "AWS::IoT::Certificate"
    }
  }
}
```

> **Note**
>
> *MyThingMyPolicy*, e *MyCertificate*sono nomi arbitrari che identificano ogni specifica di risorsa nel modello di provisioning del parco veicoli. AWS IoTnon utilizza questi nomi nelle risorse che crea dal modello. È possibile utilizzare questi nomi o sostituirli con valori che consentano di identificare ogni risorsa nel modello.

b. Crea il modello di approvvigionamento del parco veicoli dal documento relativo al modello di approvvigionamento.

- Sostituiscilo *GreengrassFleetProvisioningTemplate*con il nome del modello da creare.
- Sostituisci la descrizione del modello con una descrizione del modello.
- Sostituisci l'ARN del ruolo di provisioning con l'ARN del ruolo creato in precedenza.

Linux or Unix

```
aws iot create-provisioning-template \
  --template-name GreengrassFleetProvisioningTemplate \
  --description "A provisioning template for Greengrass core devices." \
  --provisioning-role-arn "arn:aws:iam::123456789012:role/
GreengrassFleetProvisioningRole" \
  --template-body file://greengrass-fleet-provisioning-template.json \
  --enabled
```

Windows Command Prompt (CMD)

```
aws iot create-provisioning-template ^
  --template-name GreengrassFleetProvisioningTemplate ^
  --description "A provisioning template for Greengrass core devices." ^
  --provisioning-role-arn "arn:aws:iam::123456789012:role/
GreengrassFleetProvisioningRole" ^
  --template-body file://greengrass-fleet-provisioning-template.json ^
  --enabled
```

PowerShell

```
aws iot create-provisioning-template `
  --template-name GreengrassFleetProvisioningTemplate `
  --description "A provisioning template for Greengrass core devices." `
  --provisioning-role-arn "arn:aws:iam::123456789012:role/
GreengrassFleetProvisioningRole" `
  --template-body file://greengrass-fleet-provisioning-template.json `
  --enabled
```

> **Note**
>
> Se hai creato un hook di pre-provisioning, specifica l'ARN della funzione Lambda dell'hook di pre-provisioning con l'argomento. `--pre-provisioning-hook`
>
> ```
> --pre-provisioning-hook targetArn=arn:aws:lambda:us-
> west-2:123456789012:function:GreengrassPreProvisioningHook
> ```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
    "templateArn": "arn:aws:iot:us-west-2:123456789012:provisioningtemplate/
GreengrassFleetProvisioningTemplate",
    "templateName": "GreengrassFleetProvisioningTemplate",
    "defaultVersionId": 1
}
```

Crea un certificato di richiesta di approvvigionamento e una chiave privata

I certificati di attestazione sono certificati X.509 che consentono ai dispositivi di registrarsi come AWS IoT oggetti e di recuperare un certificato di dispositivo X.509 univoco da utilizzare per le normali operazioni. Dopo aver creato un certificato di attestazione, alleghi una AWS IoT politica che consente ai dispositivi di utilizzarlo per creare certificati di dispositivo unici e fornire un modello di provisioning del parco veicoli. I dispositivi con il certificato di richiesta possono effettuare il provisioning utilizzando solo il modello di provisioning consentito nella AWS IoT politica.

In questa sezione, crei il certificato di richiesta e lo configuri per i dispositivi da utilizzare con il modello di provisioning del parco veicoli creato nella sezione precedente.

> **Important**
>
> Il provisioning dichiara che le chiavi private devono essere protette in ogni momento, anche sui dispositivi core Greengrass. Ti consigliamo di utilizzare i CloudWatch parametri e i log di Amazon per monitorare eventuali indicazioni di uso improprio, come l'uso non autorizzato del certificato di attestazione per il provisioning dei dispositivi. Se rilevi un uso improprio, disattiva il certificato di richiesta di approvvigionamento in modo che non possa essere utilizzato per

il provisioning dei dispositivi. Per ulteriori informazioni, consulta Monitoring AWS IoT nella Developer Guide. AWS IoT Core

Per aiutarti a gestire meglio il numero di dispositivi e i dispositivi che si registrano automaticamente nel tuo sistemaAccount AWS, puoi specificare un hook di pre-provisioning quando crei un modello di provisioning del parco veicoli. Un hook di pre-provisioning è una AWS Lambda funzione che convalida i parametri del modello forniti dai dispositivi durante la registrazione. Ad esempio, è possibile creare un hook di pre-provisioning che controlli l'ID di un dispositivo confrontandolo con un database per verificare che il dispositivo disponga dell'autorizzazione al provisioning. Per ulteriori informazioni, consulta Pre-provisioning hook nella Developer Guide. AWS IoT Core

Per creare un certificato di richiesta di approvvigionamento e una chiave privata

1. Crea una cartella in cui scaricare il certificato di richiesta e la chiave privata.

   ```
   mkdir claim-certs
   ```

2. Crea e salva un certificato e una chiave privata da utilizzare per il provisioning. AWS IoTfornisce certificati client firmati dall'autorità di certificazione Amazon Root (CA).

   Linux or Unix

   ```
   aws iot create-keys-and-certificate \
     --certificate-pem-outfile "claim-certs/claim.pem.crt" \
     --public-key-outfile "claim-certs/claim.public.pem.key" \
     --private-key-outfile "claim-certs/claim.private.pem.key" \
     --set-as-active
   ```

   Windows Command Prompt (CMD)

   ```
   aws iot create-keys-and-certificate ^
     --certificate-pem-outfile "claim-certs/claim.pem.crt" ^
     --public-key-outfile "claim-certs/claim.public.pem.key" ^
     --private-key-outfile "claim-certs/claim.private.pem.key" ^
     --set-as-active
   ```

PowerShell

```
aws iot create-keys-and-certificate `
  --certificate-pem-outfile "claim-certs/claim.pem.crt" `
  --public-key-outfile "claim-certs/claim.public.pem.key" `
  --private-key-outfile "claim-certs/claim.private.pem.key" `
  --set-as-active
```

La risposta contiene informazioni sul certificato, se la richiesta ha esito positivo. Salva l'ARN del certificato per utilizzarlo in seguito.

3. Crea e allega una AWS IoT policy che consenta ai dispositivi di utilizzare il certificato per creare certificati univoci per i dispositivi ed esegui il provisioning con il modello di provisioning del parco veicoli. La seguente policy consente l'accesso all'API MQTT per il provisioning dei dispositivi. Per ulteriori informazioni, consulta Device Provisioning MQTT API nella Device Provisioning nella Developer Guide. AWS IoT Core

   Esegui questa operazione:

   a. Crea un file che contenga il documento di AWS IoT policy richiesto dai dispositivi core Greengrass.

      Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

      ```
      nano greengrass-provisioning-claim-iot-policy.json
      ```

      Copiate il seguente codice JSON nel file.

      - Sostituisci ogni istanza della *regione* con quella Regione AWS in cui hai impostato il provisioning della flotta.
      - Sostituisci ogni istanza di *account-id con il tuo ID*. Account AWS
      - Sostituisci ogni istanza di *GreengrassFleetProvisioningTemplate*con il nome del modello di approvvigionamento della flotta creato nella sezione precedente.

      ```
      {
        "Version": "2012-10-17",
      ```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:topic/$aws/certificates/create/*",
        "arn:aws:iot:region:account-id:topic/$aws/provisioning-
templates/GreengrassFleetProvisioningTemplate/provision/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": [
        "arn:aws:iot:region:account-id:topicfilter/$aws/certificates/create/*",
        "arn:aws:iot:region:account-id:topicfilter/$aws/provisioning-
templates/GreengrassFleetProvisioningTemplate/provision/*"
      ]
    }
  ]
}
```

b. Crea una AWS IoT politica dal documento di policy.

- Sostituisci *GreengrassProvisioningClaimPolicy*con il nome della politica da creare.

```
aws iot create-policy --policy-name GreengrassProvisioningClaimPolicy --policy-
document file://greengrass-provisioning-claim-iot-policy.json
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "policyName": "GreengrassProvisioningClaimPolicy",
```

```
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassProvisioningClaimPolicy",
  "policyDocument": "{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
      {
        \"Effect\": \"Allow\",
        \"Action\": \"iot:Connect\",
        \"Resource\": \"*\"
      },
      {
        \"Effect\": \"Allow\",
        \"Action\": [
          \"iot:Publish\",
          \"iot:Receive\"
        ],
        \"Resource\": [
          \"arn:aws:iot:region:account-id:topic/$aws/certificates/create/*\",
          \"arn:aws:iot:region:account-id:topic/$aws/provisioning-
templates/GreengrassFleetProvisioningTemplate/provision/*\"
        ]
      },
      {
        \"Effect\": \"Allow\",
        \"Action\": \"iot:Subscribe\",
        \"Resource\": [
          \"arn:aws:iot:region:account-id:topicfilter/$aws/certificates/create/
*\",
          \"arn:aws:iot:region:account-id:topicfilter/$aws/provisioning-
templates/GreengrassFleetProvisioningTemplate/provision/*\"
        ]
      }
    ]
  }",
  "policyVersionId": "1"
}
```

4. Allega la AWS IoT politica al certificato di richiesta di approvvigionamento.

   - Sostituiscila *GreengrassProvisioningClaimPolicy*con il nome della politica da allegare.
   - Sostituisci l'ARN di destinazione con l'ARN del certificato di richiesta di approvvigionamento.

```
aws iot attach-policy --policy-name GreengrassProvisioningClaimPolicy --
target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

Il comando non produce alcun output se la richiesta ha esito positivo.

Ora disponi di un certificato di richiesta di provisioning e di una chiave privata che i dispositivi possono utilizzare per registrarsi AWS IoT e rifornirsi come dispositivi core Greengrass. È possibile incorporare il certificato di richiesta e la chiave privata nei dispositivi durante la produzione oppure copiare il certificato e la chiave sui dispositivi prima di installare il AWS IoT Greengrass software Core. Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con il provisioning AWS IoT della flotta.

Configurare il plug-in per il provisioning AWS IoT della flotta

Il plug-in per il provisioning AWS IoT della flotta fornisce i seguenti parametri di configurazione che è possibile personalizzare quando si installa il software AWS IoT Greengrass Core con il provisioning della flotta.

`rootPath`

Il percorso della cartella da utilizzare come root per il software AWS IoT Greengrass Core.

`awsRegion`

Regione AWSQuello che il plug-in Fleet Provisioning utilizza per fornire AWS le risorse.

`iotDataEndpoint`

L'endpoint di AWS IoT dati per il tuo. Account AWS

`iotCredentialEndpoint`

L'endpoint di AWS IoT credenziali per il tuo. Account AWS

`iotRoleAlias`

L'alias del AWS IoT ruolo che rimanda a un ruolo IAM per lo scambio di token. Il fornitore di AWS IoT credenziali assume questo ruolo per consentire al dispositivo principale Greengrass di interagire con i servizi. AWS Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

`provisioningTemplate`

Il modello di provisioning AWS IoT della flotta da utilizzare per il provisioning delle risorse. AWS Questo modello deve specificare quanto segue:

- Qualsiasi AWS IoT cosa, risorsa. È possibile specificare un elenco di gruppi di oggetti esistenti per distribuire i componenti su ciascun dispositivo quando è online.
- Una risorsa AWS IoT politica. Questa risorsa può definire una delle seguenti proprietà:
  - Il nome di una AWS IoT politica esistente. Se scegli questa opzione, i dispositivi principali che crei a partire da questo modello utilizzano la stessa AWS IoT politica e puoi gestirne le autorizzazioni come flotta.
  - Un documento AWS IoT di policy. Se scegli questa opzione, ogni dispositivo principale creato a partire da questo modello utilizza una AWS IoT policy unica e puoi gestire le autorizzazioni per ogni singolo dispositivo principale.
- Una risorsa AWS IoT certificata. Questa risorsa di certificato deve utilizzare il `AWS::IoT::Certificate::Id` parametro per allegare il certificato al dispositivo principale. Per ulteriori informazioni, consulta J ust-in-time provisioning nella AWS IoTDeveloper Guide.

Per ulteriori informazioni, consulta Provisioning templates nella AWS IoT CoreDeveloper Guide.

`claimCertificatePath`

Il percorso del certificato di richiesta di provisioning per il modello di provisioning specificato in. `provisioningTemplate` Per ulteriori informazioni, consulta CreateProvisioningClaim nella documentazione di riferimento dell'API AWS IoT Core.

`claimCertificatePrivateKeyPath`

Il percorso della chiave privata del certificato di richiesta di approvvigionamento per il modello di provisioning specificato in. `provisioningTemplate` Per ulteriori informazioni, consulta CreateProvisioningClaim nella documentazione di riferimento dell'API AWS IoT Core.

> **Important**
>
> Il provisioning dichiara che le chiavi private devono essere protette in ogni momento, anche sui dispositivi core Greengrass. Ti consigliamo di utilizzare i CloudWatch parametri e i log di Amazon per monitorare eventuali indicazioni di uso improprio, come l'uso non autorizzato del certificato di attestazione per il provisioning dei dispositivi. Se rilevi un uso improprio, disattiva il certificato di richiesta di approvvigionamento in modo che non

possa essere utilizzato per il provisioning dei dispositivi. Per ulteriori informazioni, consulta Monitoring AWS IoT nella Developer Guide. AWS IoT Core
Per aiutarti a gestire meglio il numero di dispositivi e i dispositivi che si registrano automaticamente nel tuo sistemaAccount AWS, puoi specificare un hook di pre-provisioning quando crei un modello di provisioning del parco veicoli. Un hook di pre-provisioning è una AWS Lambda funzione che convalida i parametri del modello forniti dai dispositivi durante la registrazione. Ad esempio, è possibile creare un hook di pre-provisioning che controlli l'ID di un dispositivo confrontandolo con un database per verificare che il dispositivo disponga dell'autorizzazione al provisioning. Per ulteriori informazioni, consulta Pre-provisioning hook nella Developer Guide. AWS IoT Core

`rootCaPath`

Il percorso verso il certificato Amazon Root Certificate Authority (CA).

`templateParameters`

(Facoltativo) La mappa dei parametri da fornire al modello di provisioning della flotta. Per ulteriori informazioni, consulta la sezione Parametri dei modelli di provisioning nella Guida per gli sviluppatori. AWS IoT Core

`deviceId`

(Facoltativo) L'identificatore del dispositivo da utilizzare come ID client quando il plug-in Fleet Provisioning crea una connessione MQTT. AWS IoT

Impostazione predefinita: un UUID casuale.

`mqttPort`

(Facoltativo) La porta da usare per le connessioni MQTT.

Impostazione predefinita: `8883`

`proxyUrl`

(Facoltativo) L'URL del server proxy nel formato`scheme://userinfo@host:port`. Per utilizzare un proxy HTTPS, è necessario utilizzare la versione 1.1.0 o successiva del plug-in Fleet Provisioning.

- `scheme`— Lo schema, che deve essere `http` o. `https`

> **Important**
>
> I dispositivi core Greengrass devono eseguire Greengrass nucleus v2.5.0 o versione successiva per utilizzare i proxy HTTPS.
> Se configuri un proxy HTTPS, devi aggiungere il certificato CA del server proxy al certificato Amazon root CA del dispositivo principale. Per ulteriori informazioni, consulta Abilita il dispositivo principale in modo che consideri attendibile un proxy HTTPS.

- `userinfo`— (Facoltativo) Le informazioni sul nome utente e sulla password. Se si specificano queste informazioni nel`url`, il dispositivo principale Greengrass ignora i `username` campi and. `password`
- `host`— Il nome host o l'indirizzo IP del server proxy.
- `port`— (Facoltativo) Il numero di porta. Se non specifichi la porta, il dispositivo principale Greengrass utilizza i seguenti valori predefiniti:
  - `http`— 80
  - `https`— 443

`proxyUserName`

(Facoltativo) Il nome utente che autentica il server proxy.

`proxyPassword`

(Facoltativo) Il nome utente che autentica il server proxy.

CSRPath

(Facoltativo) Il percorso del file di richiesta di firma del certificato (CSR) da utilizzare per creare il certificato del dispositivo da una CSR. Per ulteriori informazioni, consulta Provisioning by claim nella guida per sviluppatori. AWS IoT Core

csrPrivateKeyPercorso

(Facoltativo, `csrPath` obbligatorio se dichiarato) Il percorso della chiave privata utilizzata per generare la CSR. La chiave privata deve essere stata utilizzata per generare la CSR. Per ulteriori informazioni, consulta Provisioning by claim nella guida per AWS IoT Coresviluppatori.

AWS IoT log delle modifiche del plugin per il provisioning della flotta

La tabella seguente descrive le modifiche in ciascuna versione del plug-in AWS IoT Fleet Provisioning by claim (`aws.greengrass.FleetProvisioningByClaim`).

Versione	Modifiche
1.2.1	Correzioni di bug e miglioramenti • Risolve un problema per cui il plug-in di provisioning della flotta è offline durante l'avvio di Greengrass nucleus. Il plug-in di provisioning della flotta ora riprova a tempo indeterminato le chiamate MQTT Connect.
1.2.0	Correzioni di bug e miglioramenti • Aggiunge il supporto per il provisioning dei dispositivi tramite richiesta di firma del certificato con percorso di chiave privata configurabile. • Correzioni e miglioramenti minori.
1.1.0	Correzioni di bug e miglioramenti • Aggiunge il supporto per formati di percorsi di file aggiuntivi quando si configura il plug-in su dispositivi Windows. • Aggiunge il supporto per le configurazioni proxy di rete HTTPS. Per ulteriori informazioni, consulta Connessione alla porta 443 o tramite un proxy di rete e Abilita il dispositivo principale in modo che consideri attendibile un proxy HTTPS.
1.0.0	Versione iniziale.

Installa il software AWS IoT Greengrass Core con provisioning personalizzato delle risorse

Questa funzionalità è disponibile per la versione 2.4.0 e successive del componente Greengrass nucleus.

Il programma di installazione del software AWS IoT Greengrass Core fornisce un'interfaccia Java che è possibile implementare in un plug-in personalizzato che fornisce le risorse necessarie. AWS È possibile sviluppare un plug-in di provisioning per utilizzare certificati client X.509 personalizzati o per

eseguire passaggi di provisioning complessi che altri processi di installazione non supportano. Per ulteriori informazioni, consulta Creare i propri certificati client nella Guida per gli sviluppatori.AWS IoT Core

Per eseguire un plug-in di provisioning personalizzato quando installate il software AWS IoT Greengrass Core, create un file JAR da fornire all'installatore. Il programma di installazione esegue il plug-in e il plug-in restituisce una configurazione di provisioning che definisce AWS le risorse per il dispositivo principale Greengrass. Il programma di installazione utilizza queste informazioni per configurare il software AWS IoT Greengrass Core sul dispositivo. Per ulteriori informazioni, consulta Sviluppa plugin di provisioning personalizzati.

> **Important**
>
> Prima di scaricare il software AWS IoT Greengrass Core, verifica che il dispositivo principale soddisfi i requisiti per installare ed eseguire il software AWS IoT Greengrass Core v2.0.

Argomenti

Prerequisiti

Per installare il software AWS IoT Greengrass Core con provisioning personalizzato, è necessario disporre di quanto segue:

- Un file JAR per un plug-in di provisioning personalizzato che implementa il `DeviceIdentityInterface` Il plug-in di provisioning personalizzato deve restituire valori per ogni parametro di configurazione del sistema e del nucleo. Altrimenti, è necessario fornire tali valori nel file di configurazione durante l'installazione. Per ulteriori informazioni, consulta Sviluppa plugin di provisioning personalizzati.

Configura l'ambiente del dispositivo

Segui i passaggi di questa sezione per configurare un dispositivo Linux o Windows da utilizzare come dispositivo AWS IoT Greengrass principale.

Configura un dispositivo Linux

Per configurare un dispositivo Linux per AWS IoT Greengrass V2

1. Installa il runtime Java, necessario per l'esecuzione del software AWS IoT Greengrass Core. Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva. I seguenti comandi mostrano come installare OpenJDK sul tuo dispositivo.

   - Per le distribuzioni basate su Debian o basate su Ubuntuu:

     ```
     sudo apt install default-jdk
     ```

   - Per le distribuzioni basate su Red Hat:

     ```
     sudo yum install java-11-openjdk-devel
     ```

   - Per Amazon Linux 2:

     ```
     sudo amazon-linux-extras install java-openjdk11
     ```

   - Per Amazon Linux 2023:

     ```
     sudo dnf install java-11-amazon-corretto -y
     ```

   Al termine dell'installazione, esegui il comando seguente per verificare che Java sia in esecuzione sul tuo dispositivo Linux.

   ```
   java -version
   ```

   Il comando stampa la versione di Java in esecuzione sul dispositivo. Ad esempio, su una distribuzione basata su Debian, l'output potrebbe essere simile all'esempio seguente.

   ```
   openjdk version "11.0.9.1" 2020-11-04
   ```

```
OpenJDK Runtime Environment (build 11.0.9.1+1-post-Debian-1deb10u2)
OpenJDK 64-Bit Server VM (build 11.0.9.1+1-post-Debian-1deb10u2, mixed mode)
```

2. (Facoltativo) Crea l'utente e il gruppo di sistema predefiniti che eseguono i componenti sul dispositivo. Puoi anche scegliere di lasciare che il programma di installazione del software AWS IoT Greengrass Core crei questo utente e gruppo durante l'installazione con l'argomento `--component-default-user` installer. Per ulteriori informazioni, consulta Argomenti dell'installatore.

   ```
   sudo useradd --system --create-home ggc_user
   sudo groupadd --system ggc_group
   ```

3. Verificate che l'utente che esegue il software AWS IoT Greengrass Core (in genere`root`) sia autorizzato a funzionare `sudo` con qualsiasi utente e gruppo.

   a. Eseguite il comando seguente per aprire il `/etc/sudoers` file.

      ```
      sudo visudo
      ```

   b. Verificate che l'autorizzazione per l'utente sia simile all'esempio seguente.

      ```
      root    ALL=(ALL:ALL) ALL
      ```

4. (Facoltativo) Per eseguire funzioni Lambda containerizzate, è necessario abilitare cgroups v1 e abilitare e montare i cgroup di memoria e dispositivi. Se non intendi eseguire funzioni Lambda containerizzate, puoi saltare questo passaggio.

   Per abilitare queste opzioni di cgroups, avvia il dispositivo con i seguenti parametri del kernel Linux.

   ```
   cgroup_enable=memory cgroup_memory=1 systemd.unified_cgroup_hierarchy=0
   ```

   Per informazioni sulla visualizzazione e l'impostazione dei parametri del kernel per il tuo dispositivo, consulta la documentazione del tuo sistema operativo e del boot loader. Segui le istruzioni per impostare in modo permanente i parametri del kernel.

5. Installa tutte le altre dipendenze richieste sul tuo dispositivo come indicato dall'elenco dei requisiti in. Requisiti per il dispositivo

Configura un dispositivo Windows

> **Note**
>
> Questa funzionalità è disponibile per la versione 2.5.0 e successive del componente Greengrass nucleus.

Per configurare un dispositivo Windows per AWS IoT Greengrass V2

1. Installa il runtime Java, necessario per l'esecuzione del software AWS IoT Greengrass Core. Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva.
2. Controlla se Java è disponibile nella variabile di sistema PATH e aggiungilo in caso contrario. L' LocalSystem account esegue il software AWS IoT Greengrass Core, quindi è necessario aggiungere Java alla variabile di sistema PATH anziché alla variabile utente PATH per l'utente. Esegui questa operazione:
   a. Premi il tasto Windows per aprire il menu di avvio.
   b. Digita **`environment variables`** per cercare le opzioni di sistema dal menu di avvio.
   c. Nei risultati della ricerca del menu di avvio, scegli Modifica le variabili di ambiente di sistema per aprire la finestra delle proprietà del sistema.
   d. Scegli le variabili di ambiente... per aprire la finestra Variabili d'ambiente.
   e. In Variabili di sistema, seleziona Percorso, quindi scegli Modifica. Nella finestra Modifica variabile di ambiente, puoi visualizzare ogni percorso su una riga separata.
   f. Controlla se è presente il percorso della `bin` cartella di installazione di Java. Il percorso potrebbe essere simile all'esempio seguente.

      ```
      C:\\Program Files\\Amazon Corretto\\jdk11.0.13_8\\bin
      ```

   g. Se la `bin` cartella di installazione Java non è presente in Path, scegliete Nuovo per aggiungerla, quindi scegliete OK.
3. Aprite il prompt dei comandi di Windows (`cmd.exe`) come amministratore.
4. Crea l'utente predefinito nell' LocalSystem account sul dispositivo Windows. Sostituisci *`la password`* con una password sicura.

```
net user /add ggc_user password
```

> **Tip**
>
> A seconda della configurazione di Windows, la password dell'utente potrebbe essere impostata per scadere in date future. Per garantire che le tue applicazioni Greengrass continuino a funzionare, tieni traccia della scadenza della password e aggiornala prima che scada. Puoi anche impostare la password dell'utente in modo che non scada mai.
>
> - Per verificare la scadenza di un utente e della relativa password, esegui il comando seguente.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Per impostare la password di un utente in modo che non scada mai, esegui il comando seguente.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```
>
> - Se utilizzi Windows 10 o versioni successive in cui il wmiccomando è obsoleto, esegui il comando seguente. PowerShell
>
>   ```
>   Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
>    'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
>   ```

5. Scarica e installa l'PsExecutilità di Microsoft sul dispositivo.
6. Utilizzate l' PsExec utilità per memorizzare il nome utente e la password per l'utente predefinito nell'istanza di Credential Manager per l' LocalSystem account. Sostituisci la *password* con la password dell'utente impostata in precedenza.

   ```
   psexec -s cmd /c cmdkey /generic:ggc_user /user:ggc_user /pass:password
   ```

   Se si PsExec License Agreementapre, scegli Acceptdi accettare la licenza ed esegui il comando.

> **Note**
>
> Sui dispositivi Windows, l' LocalSystem account esegue il Greengrass nucleus ed è necessario utilizzare l' PsExec utilità per memorizzare le informazioni utente predefinite nell'account. LocalSystem L'utilizzo dell'applicazione Credential Manager archivia queste informazioni nell'account Windows dell'utente attualmente connesso, anziché nell'account. LocalSystem

Scaricate il software Core AWS IoT Greengrass

È possibile scaricare la versione più recente del software AWS IoT Greengrass Core dal seguente indirizzo:

- https://d2s8p88vqu9w66.cloudfront.net/releases/ greengrass-nucleus-latest .zip

> **Note**
>
> È possibile scaricare una versione specifica del software AWS IoT Greengrass Core dal seguente percorso. Sostituisci la *versione* con la versione da scaricare.
>
> ```
> https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip
> ```

Per scaricare il software AWS IoT Greengrass Core

1. Sul dispositivo principale, scaricate il software AWS IoT Greengrass Core in un file denominato`greengrass-nucleus-latest.zip`.

   Linux or Unix

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
   latest.zip > greengrass-nucleus-latest.zip
   ```

Windows Command Prompt (CMD)

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
latest.zip > greengrass-nucleus-latest.zip
```

PowerShell

```
iwr -Uri https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-
latest.zip -OutFile greengrass-nucleus-latest.zip
```

Scaricando questo software accetti l'Accordo di licenza del software Greengrass Core.

2. (Facoltativo) Per verificare la firma del software Greengrass nucleus

> **Note**
>
> Questa funzionalità è disponibile con Greengrass nucleus versione 2.9.5 e successive.

a. Usa il seguente comando per verificare la firma del tuo artefatto Greengrass nucleus:

Linux or Unix

```
jarsigner -verify -certs -verbose greengrass-nucleus-latest.zip
```

Windows Command Prompt (CMD)

Il nome del file potrebbe avere un aspetto diverso a seconda della versione di JDK installata. *jdk17.0.6_10*Sostituiscilo con la versione JDK che hai installato.

```
"C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe" -
verify -certs -verbose greengrass-nucleus-latest.zip
```

PowerShell

Il nome del file potrebbe avere un aspetto diverso a seconda della versione di JDK installata. *jdk17.0.6_10*Sostituiscilo con la versione JDK che hai installato.

```
'C:\\Program Files\\Amazon Corretto\\jdk17.0.6_10\\bin\\jarsigner.exe' -
verify -certs -verbose greengrass-nucleus-latest.zip
```

b. L'`jarsigner`invocazione produce un output che indica i risultati della verifica.

   i. Se il file zip Greengrass nucleus è firmato, l'output contiene la seguente dichiarazione:

   ```
   jar verified.
   ```

   ii. Se il file zip Greengrass nucleus non è firmato, l'output contiene la seguente dichiarazione:

   ```
   jar is unsigned.
   ```

c. Se hai fornito l'`-certs`opzione Jarsigner insieme alle `-verbose` opzioni `-verify` e, l'output include anche informazioni dettagliate sul certificato del firmatario.

3. Decomprimi il software AWS IoT Greengrass Core in una cartella sul tuo dispositivo. Sostituiscilo *`GreengrassInstaller`*con la cartella che desideri utilizzare.

   Linux or Unix

   ```
   unzip greengrass-nucleus-latest.zip -d GreengrassInstaller && rm greengrass-
   nucleus-latest.zip
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir GreengrassInstaller && tar -xf greengrass-nucleus-latest.zip -
   C GreengrassInstaller && del greengrass-nucleus-latest.zip
   ```

   PowerShell

   ```
   Expand-Archive -Path greengrass-nucleus-latest.zip -DestinationPath .\
   \GreengrassInstaller
   rm greengrass-nucleus-latest.zip
   ```

4. (Facoltativo) Eseguite il comando seguente per visualizzare la versione del software AWS IoT Greengrass Core.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

> **Important**
>
> Se installi una versione del nucleo Greengrass precedente alla v2.4.0, non rimuovere questa cartella dopo aver installato il software Core. AWS IoT Greengrass Il software AWS IoT Greengrass Core utilizza i file in questa cartella per l'esecuzione.
> Se hai scaricato la versione più recente del software, installi la versione 2.4.0 o successiva e puoi rimuovere questa cartella dopo aver installato il software AWS IoT Greengrass Core.

Installa il software Core AWS IoT Greengrass

Esegui il programma di installazione con argomenti che specificano le seguenti azioni:

- Effettua l'installazione da un file di configurazione parziale che specifica di utilizzare il plug-in di provisioning personalizzato per il provisioning delle risorse. AWS Il software AWS IoT Greengrass Core utilizza un file di configurazione che specifica la configurazione di ogni componente Greengrass sul dispositivo. Il programma di installazione crea un file di configurazione completo a partire dal file di configurazione parziale fornito dall'utente e dalle AWS risorse create dal plug-in di provisioning personalizzato.
- Specificate di utilizzare l'utente `ggc_user` del sistema per eseguire i componenti software sul dispositivo principale. Sui dispositivi Linux, questo comando specifica anche di utilizzare il gruppo di `ggc_group` sistema e il programma di installazione crea automaticamente l'utente e il gruppo di sistema.
- Configura il software AWS IoT Greengrass Core come servizio di sistema che viene eseguito all'avvio. Sui dispositivi Linux, ciò richiede il sistema di inizializzazione Systemd.

> **Important**
>
> Sui dispositivi Windows core, è necessario configurare il software AWS IoT Greengrass Core come servizio di sistema.

Per ulteriori informazioni sugli argomenti che è possibile specificare, vedereArgomenti dell'installatore.

> **Note**
>
> Se utilizzi AWS IoT Greengrass un dispositivo con memoria limitata, puoi controllare la quantità di memoria utilizzata dal software AWS IoT Greengrass Core. Per controllare l'allocazione della memoria, è possibile impostare le opzioni relative alla dimensione dell'heap JVM nel parametro di `jvm0ptions` configurazione del componente nucleus. Per ulteriori informazioni, consulta Controlla l'allocazione della memoria con le opzioni JVM.

Per installare il software Core (Linux AWS IoT Greengrass)

1. Controlla la versione del software AWS IoT Greengrass Core.

   - Sostituisci *GreengrassInstaller*con il percorso della cartella che contiene il software.

   ```
   java -jar ./GreengrassInstaller/lib/Greengrass.jar --version
   ```

2. Utilizzate un editor di testo per creare un file di configurazione denominato `config.yaml` da fornire all'installatore.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

   ```
   nano GreengrassInstaller/config.yaml
   ```

   Copiate il seguente contenuto YAML nel file.

   ```
   ---
   system:
     rootpath: "/greengrass/v2"
     # The following values are optional. Return them from the provisioning plugin or
    set them here.
     # certificateFilePath: ""
     # privateKeyPath: ""
     # rootCaPath: ""
     # thingName: ""
   services:
     aws.greengrass.Nucleus:
       version: "2.12.3"
   ```

```
    configuration:
      # The following values are optional. Return them from the provisioning plugin
or set them here.
      # awsRegion: ""
      # iotRoleAlias: ""
      # iotDataEndpoint: ""
      # iotCredEndpoint: ""
  com.example.CustomProvisioning:
    configuration:
      # You can specify configuration parameters to provide to your plugin.
      # pluginParameter: ""
```

Successivamente, esegui queste operazioni:

- Sostituisci *2.12.3* con la versione del software Core. AWS IoT Greengrass
- Sostituisci ogni istanza di */greengrass/v2* con la cartella principale di Greengrass.
- (Facoltativo) Specificate i valori di configurazione del sistema e del nucleo. È necessario impostare questi valori se il plug-in di provisioning non li fornisce.
- (Facoltativo) Specificate i parametri di configurazione da fornire al vostro plugin di provisioning.

> **Note**
>
> In questo file di configurazione, è possibile personalizzare altre opzioni di configurazione, come le porte e il proxy di rete da utilizzare, come illustrato nell'esempio seguente. Per ulteriori informazioni, vedere Greengrass nucleus configuration.
>
> ```
> ---
> system:
>   rootpath: "/greengrass/v2"
>   # The following values are optional. Return them from the provisioning
>  plugin or set them here.
>   # certificateFilePath: ""
>   # privateKeyPath: ""
>   # rootCaPath: ""
>   # thingName: ""
> services:
>   aws.greengrass.Nucleus:
>     version: "2.12.3"
>     configuration:
>       mqtt:
> ```

```
        port: 443
      greengrassDataPlanePort: 443
      networkProxy:
        noProxyAddresses: "http://192.168.0.1,www.example.com"
        proxy:
          url: "http://my-proxy-server:1100"
          username: "Mary_Major"
          password: "pass@word1357"
      # The following values are optional. Return them from the provisioning
 plugin or set them here.
      # awsRegion: ""
      # iotRoleAlias: ""
      # iotDataEndpoint: ""
      # iotCredEndpoint: ""
  com.example.CustomProvisioning:
    configuration:
      # You can specify configuration parameters to provide to your plugin.
      # pluginParameter: ""
```

3. Eseguire il programma di installazione. Specificate `--trusted-plugin` di fornire il plug-in di provisioning personalizzato e specificate di `--init-config` fornire il file di configurazione.

   - Sostituire */greengrass/v2* o *C:\greengrass\v2* con la cartella principale di Greengrass.
   - Sostituisci ogni istanza di *GreengrassInstaller*con la cartella in cui hai decompresso il programma di installazione.
   - Sostituisci il percorso del file JAR del plug-in di provisioning personalizzato con il percorso del file JAR del plug-in.

   Linux or Unix

   ```
   sudo -E java -Droot="/greengrass/v2" -Dlog.store=FILE \
     -jar ./GreengrassInstaller/lib/Greengrass.jar \
     --trusted-plugin /path/to/com.example.CustomProvisioning.jar \
     --init-config ./GreengrassInstaller/config.yaml \
     --component-default-user ggc_user:ggc_group \
     --setup-system-service true
   ```

   Windows Command Prompt (CMD)

   ```
   java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" ^
   ```

```
  -jar ./GreengrassInstaller/lib/Greengrass.jar ^
  --trusted-plugin /path/to/com.example.CustomProvisioning.jar ^
  --init-config ./GreengrassInstaller/config.yaml ^
  --component-default-user ggc_user ^
  --setup-system-service true
```

PowerShell

```
java -Droot="C:\greengrass\v2" "-Dlog.store=FILE" `
  -jar ./GreengrassInstaller/lib/Greengrass.jar `
  --trusted-plugin /path/to/com.example.CustomProvisioning.jar `
  --init-config ./GreengrassInstaller/config.yaml `
  --component-default-user ggc_user `
  --setup-system-service true
```

> **Important**
>
> Nei dispositivi Windows core, è necessario specificare `--setup-system-service true` di configurare il software AWS IoT Greengrass Core come servizio di sistema.

Se si specifica`--setup-system-service true`, il programma di installazione stampa `Successfully set up Nucleus as a system service` se ha configurato ed eseguito il software come servizio di sistema. Altrimenti, il programma di installazione non emette alcun messaggio se installa il software correttamente.

> **Note**
>
> Non è possibile utilizzare l'`deploy-dev-tools`argomento per distribuire strumenti di sviluppo locali quando si esegue il programma di installazione senza l'argomento. `--provision true` Per informazioni sulla distribuzione della CLI Greengrass direttamente sul dispositivo, consulta. Interfaccia a riga di comando Greengrass

4. Verifica l'installazione visualizzando i file nella cartella principale.

   Linux or Unix

   ```
   ls /greengrass/v2
   ```

Windows Command Prompt (CMD)

```
dir C:\greengrass\v2
```

PowerShell

```
ls C:\greengrass\v2
```

Se l'installazione è riuscita, la cartella principale contiene diverse cartelle, ad esempio `configpackages`, `elogs`.

Se avete installato il software AWS IoT Greengrass Core come servizio di sistema, il programma di installazione esegue il software automaticamente. In caso contrario, è necessario eseguire il software manualmente. Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core.

Per ulteriori informazioni su come configurare e utilizzare il software AWS IoT Greengrass, consulta quanto segue:

- Configurare il software AWS IoT Greengrass Core
- Sviluppa AWS IoT Greengrass componenti
- Implementazione AWS IoT Greengrass dei componenti sui dispositivi
- Interfaccia a riga di comando Greengrass

Sviluppa plugin di provisioning personalizzati

Per sviluppare un plugin di provisioning personalizzato, creare una classe Java che implementa`com.aws.greengrass.provisioning.DeviceIdentityInterface`interfaccia. Puoi includere il file JAR del nucleo Greengrass nel tuo progetto per accedere a questa interfaccia e alle sue classi. Questa interfaccia definisce un metodo che inserisce una configurazione di plugin e genera una configurazione di provisioning. La configurazione di provisioning definisce le configurazioni per il sistema e ilComponente nucleo Greengrass. LaAWS IoT GreengrassIl programma di installazione del software di base utilizza questa configurazione di provisioning per configurare ilAWS IoT GreengrassSoftware di base su un dispositivo.

Dopo aver sviluppato un plugin di provisioning personalizzato, crealo come file JAR che puoi fornire alAWS IoT GreengrassInstaller software di base per eseguire il plugin durante l'installazione. Il

programma di installazione esegue il plugin di provisioning personalizzato nella stessa JVM utilizzata dal programma di installazione, in modo da poter creare un JAR che contiene solo il codice del plugin.

> **Note**
>
> LaAWS IoTplugin di Provisioning del parco ISTanzeimplementare il`DeviceIdentityInterface`per utilizzare il provisioning del parco ISTanze durante l'installazione. Il plugin per il provisioning della flotta è open source, quindi puoi esplorare il codice sorgente per vedere un esempio di come utilizzare l'interfaccia del plugin di provisioning. Per ulteriori informazioni, consulta la .AWS IoTplugin di Provisioning del parco ISTanzesu GitHub

Argomenti

- Requisiti
- Implementare il DeviceIdentityInterface interfaccia

Requisiti

Per sviluppare un plugin di provisioning personalizzato, è necessario creare una classe Java che soddisfi i seguenti requisiti:

- Usa il`com.aws.greengrass`pacchetto o un pacchetto all'interno del`com.aws.greengrass`pacchetto.
- Ha un costruttore senza argomenti.
- Implementazione del`DeviceIdentityInterface`interfaccia. Per ulteriori informazioni, consulta la pagina Implementare il DeviceIdentityInterface interfaccia .

Implementare il DeviceIdentityInterface interfaccia

Per utilizzare il plugin`com.aws.greengrass.provisioning.DeviceIdentityInterface`interfaccia nel tuo plugin personalizzato, aggiungi il nucleo Greengrass come dipendenza dal tuo progetto.

Per utilizzare il plugin DeviceIdentityInterface in un progetto di plugin di provisioning personalizzato

- È possibile aggiungere il file JAR del nucleo Greengrass come libreria o aggiungere il nucleo Greengrass come dipendenza Maven. Completa una delle seguenti operazioni:
  - Per aggiungere il file JAR del nucleo Greengrass come libreria, scarica il file JARAWS IoT GreengrassSoftware di base, che contiene il nucleo di Greengrass JAR. Puoi scaricare la versione più recente diAWS IoT GreengrassSoftware Core dalla posizione seguente:
    - https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-nucleus-latest.zip

    Puoi trovare il file JAR del nucleo Greengrass (`Greengrass.jar`) nel`lib`cartella nel file ZIP. Aggiungi questo file JAR al tuo progetto.
  - Per consumare il nucleo Greengrass in un progetto Maven, aggiungi una dipendenza dal`nucleus`artefatto nel`com.aws.greengrass`gruppo. Devi anche aggiungere il`greengrass-common`repository, perché il nucleo Greengrass non è disponibile nel Maven Central Repository.

```
<project ...>
    ...
    <repositories>
        <repository>
            <id>greengrass-common</id>
            <name>greengrass common</name>
            <url>https://d2jrmugq4soldf.cloudfront.net/snapshots</url>
        </repository>
    </repositories>
    ...
    <dependencies>
        <dependency>
            <groupId>com.aws.greengrass</groupId>
            <artifactId>nucleus</artifactId>
            <version>2.5.0-SNAPSHOT</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>
</project>
```

Interfaccia DeviceIdentityInterface

La `com.aws.greengrass.provisioning.DeviceIdentityInterface` ha la forma seguente.

> **Note**
>
> Puoi anche esplorare queste lezioni nel pacchetto com.aws.greengrass.provisioning del Codice sorgente del nucleo Greengrass su GitHub

```
public interface com.aws.greengrass.provisioning.DeviceIdentityInterface {
    ProvisionConfiguration updateIdentityConfiguration(ProvisionContext context)
            throws RetryableProvisioningException, InterruptedException;

    // Return the name of the plugin.
    String name();
}

com.aws.greengrass.provisioning.ProvisionConfiguration {
    SystemConfiguration systemConfiguration;
    NucleusConfiguration nucleusConfiguration
}

com.aws.greengrass.provisioning.ProvisionConfiguration.SystemConfiguration {
    String certificateFilePath;
    String privateKeyPath;
    String rootCAPath;
    String thingName;
}

com.aws.greengrass.provisioning.ProvisionConfiguration.NucleusConfiguration {
    String awsRegion;
    String iotCredentialsEndpoint;
    String iotDataEndpoint;
    String iotRoleAlias;
}

com.aws.greengrass.provisioning.ProvisioningContext {
    Map<String, Object> parameterMap;
    String provisioningPolicy;  // The policy is always "PROVISION_IF_NOT_PROVISIONED".
}
```

```
com.aws.greengrass.provisioning.exceptions.RetryableProvisioningException {}
```

Ciascun valore di configurazione nella`SystemConfiguration`e`NucleusConfiguration`è necessario per installare il*AWS IoT Greengrass*Software di base, ma è possibile restituire`null`. Se restituisce il plugin di provisioning personalizzato`null`per qualsiasi valore di configurazione, è necessario fornire tale valore nella configurazione del sistema o del nucleo quando si crea il`config.yaml`file da fornire al*AWS IoT Greengrass*Programma di installazione del software core. Se il plugin di provisioning personalizzato restituisce un valore non nullo per un'opzione definita anche in`config.yaml`, quindi il programma di installazione sostituisce il valore in`config.yaml`con il valore restituito dal plugin.

Argomenti dell'installatore

Il software AWS IoT Greengrass Core include un programma di installazione che configura il software e fornisce le AWS risorse necessarie per il funzionamento del dispositivo principale Greengrass. Il programma di installazione include i seguenti argomenti che è possibile specificare per configurare l'installazione:

`-h, --help`

(Facoltativo) Mostra le informazioni di aiuto dell'installatore.

`--version`

(Facoltativo) Mostra la versione del software AWS IoT Greengrass Core.

`-Droot`

(Facoltativo) Il percorso della cartella da utilizzare come root per il software AWS IoT Greengrass Core.

> **Note**
>
> Questo argomento imposta una proprietà JVM, quindi è necessario specificarla prima di eseguire `-jar` il programma di installazione. Ad esempio, specifica `java -Droot="/greengrass/v2" -jar /path/to/Greengrass.jar`.

Impostazione predefinita:

- Linux: `~/.greengrass`

- Windows: `%USERPROFILE%/.greengrass`

`-ar`, `--aws-region`

Regione AWSQuello che il software AWS IoT Greengrass Core utilizza per recuperare o creare le risorse richieste. AWS

`-p`, `--provision`

(Facoltativo) È possibile registrare questo dispositivo come AWS IoT oggetto e fornire le AWS risorse richieste dal dispositivo principale. Se si specifica`true`, il software AWS IoT Greengrass Core fornisce qualsiasi AWS IoT oggetto, (facoltativo) un AWS IoT gruppo di oggetti, un ruolo IAM e un alias di AWS IoT ruolo.

Impostazione predefinita: `false`

`-tn`, `--thing-name`

(Facoltativo) Il nome dell'AWS IoTelemento che registrate come dispositivo principale. Se l'oggetto con questo nome non esiste nel tuo computerAccount AWS, il software AWS IoT Greengrass Core lo crea.

> **Note**
>
> Il nome dell'oggetto non può contenere i due punti (`:`).

È necessario specificare `--provision true` se applicare questo argomento.

Impostazione predefinita: `GreengrassV2IotThing_` più un UUID casuale.

`-tgn`, `--thing-group-name`

(Facoltativo) Il nome del AWS IoT gruppo di oggetti a cui aggiungi l'elemento del AWS IoT dispositivo principale. Se una distribuzione ha come target questo gruppo di oggetti, il dispositivo principale riceve tale distribuzione quando si connette aAWS IoT Greengrass. Se il gruppo di oggetti con questo nome non esiste nel tuo computerAccount AWS, il software AWS IoT Greengrass Core lo crea.

> **Note**
>
> Il nome del gruppo di cose non può contenere i due punti (`:`).

È necessario specificare `--provision true` se applicare questo argomento.

`-tpn`, `--thing-policy-name`

Questa funzionalità è disponibile per la versione 2.4.0 e successive del componente Greengrass nucleus.

(Facoltativo) Il nome della AWS IoT policy da allegare al thing certificate di questo dispositivo principale. AWS IoT Se la AWS IoT politica con questo nome non esiste nel tuo computerAccount AWS, il software AWS IoT Greengrass Core la crea.

Il software AWS IoT Greengrass Core crea una AWS IoT politica permissiva per impostazione predefinita. È possibile definire l'ambito di questa politica o creare una politica personalizzata in cui limitare le autorizzazioni per il proprio caso d'uso. Per ulteriori informazioni, consulta AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali.

È necessario specificare `--provision true` se applicare questo argomento.

Impostazione predefinita: `GreengrassV2IoTThingPolicy`

`-trn`, `--tes-role-name`

(Facoltativo) Il nome del ruolo IAM da utilizzare per acquisire AWS credenziali che consentono al dispositivo principale di interagire con AWS i servizi. Se il ruolo con questo nome non esiste nel tuoAccount AWS, il software AWS IoT Greengrass Core lo crea con la `GreengrassV2TokenExchangeRoleAccess` policy. Questo ruolo non ha accesso ai tuoi bucket S3 dove ospiti gli artefatti dei componenti. Pertanto, è necessario aggiungere le autorizzazioni ai bucket e agli oggetti S3 degli artefatti quando si crea un componente. Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

È necessario specificare se applicare questo argomento. `--provision true`

Impostazione predefinita: `GreengrassV2TokenExchangeRole`

`-tra`, `--tes-role-alias-name`

(Facoltativo) Il nome dell'alias del AWS IoT ruolo che punta al ruolo IAM che fornisce AWS le credenziali per questo dispositivo principale. Se l'alias del ruolo con questo nome non esiste nel tuo computerAccount AWS, il software AWS IoT Greengrass Core lo crea e lo indirizza al ruolo IAM da te specificato.

È necessario specificare `--provision true` se applicare questo argomento.

Impostazione predefinita: `GreengrassV2TokenExchangeRoleAlias`

`-ss`, `--setup-system-service`

(Facoltativo) È possibile configurare il software AWS IoT Greengrass Core come servizio di sistema che viene eseguito all'avvio del dispositivo. Il nome del servizio di sistema è`greengrass`. Per ulteriori informazioni, consulta Configurare il nucleo Greengrass come servizio di sistema.

Nei sistemi operativi Linux, questo argomento richiede che il sistema init systemd sia disponibile sul dispositivo.

> **Important**
>
> Nei dispositivi Windows core, è necessario configurare il software AWS IoT Greengrass Core come servizio di sistema.

Impostazione predefinita: `false`

`-u`, `--component-default-user`

Il nome o l'ID dell'utente utilizzato dal software AWS IoT Greengrass Core per eseguire i componenti. Ad esempio, puoi specificare **`ggc_user`**. Questo valore è richiesto quando si esegue il programma di installazione su sistemi operativi Windows.

Nei sistemi operativi Linux, è anche possibile specificare facoltativamente il gruppo. Specificare l'utente e il gruppo separati da due punti. Ad esempio, **`ggc_user:ggc_group`**.

Le seguenti considerazioni aggiuntive si applicano ai sistemi operativi Linux:

- Se si esegue come root, l'utente predefinito del componente è l'utente definito nel file di configurazione. Se il file di configurazione non definisce un utente, il valore predefinito è. `ggc_user:ggc_group` Se esistono `ggc_user` o `ggc_group` non esistono, il software li crea.
- Se si esegue come utente non root, il software AWS IoT Greengrass Core utilizza quell'utente per eseguire i componenti.
- Se non specificate un gruppo, il software AWS IoT Greengrass Core utilizza il gruppo principale dell'utente del sistema.

Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti.

`-d`, `--deploy-dev-tools`

(Facoltativo) È possibile scaricare e distribuire il componente Greengrass CLI su questo dispositivo principale. È possibile utilizzare questo strumento per sviluppare ed eseguire il debug di componenti su questo dispositivo principale.

> **Important**
>
> Si consiglia di utilizzare questo componente solo in ambienti di sviluppo, non in ambienti di produzione. Questo componente fornisce l'accesso a informazioni e operazioni che in genere non sono necessarie in un ambiente di produzione. Segui il principio del privilegio minimo distribuendo questo componente solo sui dispositivi principali dove ne hai bisogno.

È necessario specificare se `--provision true` applicare questo argomento.

Impostazione predefinita: `false`

`-init`, `--init-config`

(Facoltativo) Il percorso del file di configurazione da utilizzare per installare il software AWS IoT Greengrass Core. È possibile utilizzare questa opzione per configurare nuovi dispositivi principali con una configurazione di nucleo specifica, ad esempio.

> **Important**
>
> Il file di configurazione specificato si fonde con il file di configurazione esistente sul dispositivo principale. Ciò include i componenti e le configurazioni dei componenti sul dispositivo principale. È consigliabile che il file di configurazione elenchi solo le configurazioni che si sta tentando di modificare.

`-tp`, `--trusted-plugin`

(Facoltativo) Il percorso di un file JAR da caricare come plugin affidabile. Utilizzate questa opzione per fornire i file JAR del plug-in di provisioning, ad esempio per l'installazione con fleet provisioning o il provisioning personalizzato, oppure per l'installazione con la chiave privata e il certificato in un modulo di sicurezza hardware.

`-s`, `--start`

(Facoltativo) È possibile avviare il software AWS IoT Greengrass Core dopo l'installazione e, facoltativamente, effettuare il provisioning delle risorse.

Impostazione predefinita: `true`

Esegui il software AWS IoT Greengrass Core

Dopo aver installato il software AWS IoT Greengrass Core, eseguilo per connettere il dispositivo aAWS IoT Greengrass.

Quando installi il software AWS IoT Greengrass Core, puoi specificare se installarlo come servizio di sistema con systemd. Se scegliete questa opzione, il programma di installazione esegue il software automaticamente e lo configura per l'esecuzione all'avvio del dispositivo.

> **Important**
>
> Nei dispositivi Windows Core, è necessario configurare il software AWS IoT Greengrass Core come servizio di sistema.

Argomenti

- Verificate se il software AWS IoT Greengrass Core funziona come servizio di sistema
- Esegui il software AWS IoT Greengrass Core come servizio di sistema
- Esegui il software AWS IoT Greengrass Core senza un servizio di sistema

Verificate se il software AWS IoT Greengrass Core funziona come servizio di sistema

Quando installi il software AWS IoT Greengrass Core, puoi specificare l'`--setup-system-service true`argomento per installare il software AWS IoT Greengrass Core come servizio di sistema. I dispositivi Linux richiedono il sistema systemd init per configurare il software AWS IoT Greengrass Core come servizio di sistema. Se si utilizza questa opzione, il programma di installazione esegue il software automaticamente e lo configura per l'esecuzione all'avvio del dispositivo. Il programma di installazione emette il seguente messaggio se installa correttamente il software AWS IoT Greengrass Core come servizio di sistema.

```
Successfully set up Nucleus as a system service
```

Se in precedenza avete installato il software AWS IoT Greengrass Core e non avete l'output del programma di installazione, potete verificare se il software è installato come servizio di sistema.

Per verificare se il software AWS IoT Greengrass Core è installato come servizio di sistema

- Esegui il seguente comando per verificare lo stato del servizio di sistema Greengrass.

  Linux or Unix (systemd)

  ```
  sudo systemctl status greengrass.service
  ```

  La risposta è simile all'esempio seguente se il software AWS IoT Greengrass Core è installato come servizio di sistema e attivo.

  ```
  # greengrass.service - Greengrass Core
     Loaded: loaded (/etc/systemd/system/greengrass.service; enabled; vendor
   preset: disabled)
     Active: active (running) since Thu 2021-02-11 01:33:44 UTC; 4 days ago
   Main PID: 16107 (sh)
     CGroup: /system.slice/greengrass.service
             ##16107 /bin/sh /greengrass/v2/alts/current/distro/bin/loader
             ##16111 java -Dlog.store=FILE -Droot=/greengrass/v2 -jar /greengrass/
  v2/alts/current/distro/lib/Greengrass...
  ```

  Se `systemctl` o `greengrass.service` non viene trovato, il software AWS IoT Greengrass Core non viene installato come servizio di sistema. Per eseguire il software, consultaEsegui il software AWS IoT Greengrass Core senza un servizio di sistema.

  Windows Command Prompt (CMD)

  ```
  sc query greengrass
  ```

  La risposta è simile all'esempio seguente se il software AWS IoT Greengrass Core è installato come servizio Windows e attivo.

  ```
  SERVICE_NAME: greengrass
          TYPE               : 10  WIN32_OWN_PROCESS
          STATE              : 4  RUNNING
  ```

```
                                (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE    : 0  (0x0)
        SERVICE_EXIT_CODE  : 0  (0x0)
        CHECKPOINT         : 0x0
        WAIT_HINT          : 0x0
```

PowerShell

```
Get-Service greengrass
```

La risposta è simile all'esempio seguente se il software AWS IoT Greengrass Core è installato come servizio Windows e attivo.

```
Status   Name               DisplayName
------   ----               -----------
Running  greengrass         greengrass
```

Esegui il software AWS IoT Greengrass Core come servizio di sistema

Se il software AWS IoT Greengrass Core è installato come servizio di sistema, è possibile utilizzare il gestore dei servizi di sistema per avviare, arrestare e gestire il software. Per ulteriori informazioni, consulta Configurare il nucleo Greengrass come servizio di sistema.

Per eseguire il software AWS IoT Greengrass Core

- Eseguire il comando seguente per avviare il client AWS IoT Greengrass Core.

  Linux or Unix (systemd)

  ```
  sudo systemctl start greengrass.service
  ```

  Windows Command Prompt (CMD)

  ```
  sc start greengrass
  ```

  PowerShell

  ```
  Start-Service greengrass
  ```

Esegui il software AWS IoT Greengrass Core senza un servizio di sistema

Sui dispositivi Linux core, se il software AWS IoT Greengrass Core non è installato come servizio di sistema, puoi eseguire lo script di caricamento del software per eseguire il software.

Per eseguire il software AWS IoT Greengrass Core senza un servizio di sistema

- Eseguire il comando seguente per avviare il client AWS IoT Greengrass Core. Se si esegue questo comando in un terminale, è necessario mantenere aperta la sessione del terminale per mantenere in esecuzione il software AWS IoT Greengrass Core.
  - Sostituisci */greengrass/v2* o *C:\greengrass\v2* con la cartella principale Greengrass che usi.

```
sudo /greengrass/v2/alts/current/distro/bin/loader
```

Il software stampa il seguente messaggio se viene avviato correttamente.

```
Launched Nucleus successfully.
```

Esegui il software AWS IoT Greengrass Core in un contenitore Docker

AWS IoT Greengrass può essere configurato per l'esecuzione in un contenitore Docker. Docker è una piattaforma che fornisce gli strumenti per creare, eseguire, testare e distribuire applicazioni basate su contenitori Linux. Quando esegui un'immagine AWS IoT Greengrass Docker, puoi scegliere se fornire AWS le tue credenziali al contenitore Docker e consentire al programma di installazione del software AWS IoT Greengrass Core di fornire automaticamente le risorse necessarie per il AWS funzionamento di un dispositivo core Greengrass. Se non desideri fornire AWS le credenziali, puoi effettuare manualmente il provisioning AWS delle risorse ed eseguire il software AWS IoT Greengrass Core nel contenitore Docker.

Argomenti

- Piattaforme supportate e requisiti
- AWS IoT Greengrass Download del software Docker

- Scegli come effettuare il provisioning delle risorse AWS
- Crea l'immagine del AWS IoT Greengrass contenitore da un Dockerfile
- Esegui AWS IoT Greengrass in un contenitore Docker con provisioning automatico delle risorse
- Esegui AWS IoT Greengrass in un contenitore Docker con provisioning manuale delle risorse
- Risoluzione dei problemi di AWS IoT Greengrass in un container Docker

Piattaforme supportate e requisiti

I computer host devono soddisfare i seguenti requisiti minimi per installare ed eseguire il software AWS IoT Greengrass Core in un contenitore Docker:

- Un sistema operativo basato su Linux con una connessione Internet.
- Docker Engine versione 18.09 o successiva.
- (Facoltativo) Docker Compose versione 1.22 o successiva. Docker Compose è necessario solo se si desidera utilizzare la CLI Docker Compose per eseguire le immagini Docker.

Per eseguire i componenti della funzione Lambda all'interno del contenitore Docker, è necessario configurare il contenitore per soddisfare requisiti aggiuntivi. Per ulteriori informazioni, consulta Requisiti della funzione Lambda.

Esegui i componenti in modalità processo

AWS IoT Greengrass non supporta l'esecuzione di funzioni Lambda o componenti AWS forniti in un ambiente di runtime isolato all'interno del AWS IoT Greengrass contenitore Docker. È necessario eseguire questi componenti in modalità processo senza alcun isolamento.

Quando configuri un componente della funzione Lambda, imposta la modalità di isolamento su Nessun contenitore. Per ulteriori informazioni, consulta Esegui AWS Lambda funzioni.

Quando distribuisci uno dei seguenti componenti AWS forniti, aggiorna la configurazione di ogni componente su cui impostare il `containerMode` parametro. `NoContainer` Per ulteriori informazioni sugli aggiornamenti della configurazione, vedere. Aggiornamento delle configurazioni dei componenti

- CloudWatch metriche
- Device Defender
- Firehose

- Adattatore di protocollo Modbus-RTU
- Amazon SNS

AWS IoT Greengrass Download del software Docker

AWS IoT Greengrass fornisce un Dockerfile per creare un'immagine del contenitore con software AWS IoT Greengrass Core e dipendenze installati su un'immagine base Amazon Linux 2 (x86_64). Puoi modificare l'immagine di base nel Dockerfile per eseguirla su un'architettura di piattaforma diversa. AWS IoT Greengrass

Scarica il pacchetto Dockerfile da. GitHub

Il Dockerfile utilizza una versione precedente di Greengrass. È necessario aggiornare il file per utilizzare la versione di Greengrass desiderata. Per informazioni sulla creazione dell'immagine del AWS IoT Greengrass contenitore dal Dockerfile, consulta. Crea l'immagine del AWS IoT Greengrass contenitore da un Dockerfile

Scegli come effettuare il provisioning delle risorse AWS

Quando installi il software AWS IoT Greengrass Core in un contenitore Docker, puoi scegliere se effettuare il provisioning automatico AWS delle risorse necessarie al funzionamento di un dispositivo core Greengrass o utilizzare le risorse che effettui il provisioning manuale.

- Provisioning automatico delle risorse: il programma di installazione esegue il provisioning dell' AWS IoT AWS IoT oggetto, del gruppo di oggetti, del ruolo IAM e dell'alias del AWS IoT ruolo quando si esegue l'immagine del AWS IoT Greengrass contenitore per la prima volta. Il programma di installazione può anche distribuire gli strumenti di sviluppo locali sul dispositivo principale, in modo da poter utilizzare il dispositivo per sviluppare e testare componenti software personalizzati. Per effettuare automaticamente il provisioning di queste risorse, è necessario fornire AWS le credenziali come variabili di ambiente all'immagine Docker.

  Per utilizzare il provisioning automatico, è necessario impostare la variabile di ambiente Docker `PROVISION=true` e montare un file di credenziali per fornire le AWS credenziali al contenitore.
- Fornitura manuale delle risorse: se non desideri fornire AWS credenziali al contenitore, puoi effettuare il provisioning manuale AWS delle risorse prima di eseguire l'immagine del contenitore. AWS IoT Greengrass È necessario creare un file di configurazione per fornire informazioni su queste risorse al programma di installazione del software AWS IoT Greengrass Core all'interno del contenitore Docker.

Per utilizzare il provisioning manuale, è necessario impostare la variabile di ambiente Docker. `PROVISION=false` Il provisioning manuale è l'opzione predefinita.

Per ulteriori informazioni, consulta Crea l'immagine del AWS IoT Greengrass contenitore da un Dockerfile.

Crea l'immagine del AWS IoT Greengrass contenitore da un Dockerfile

AWSfornisce un Dockerfile che puoi scaricare e utilizzare per eseguire AWS IoT Greengrass il software Core in un contenitore Docker. I Dockerfile contengono il codice sorgente per la creazione di immagini dei container. AWS IoT Greengrass

Prima di creare un'immagine del AWS IoT Greengrass contenitore, devi configurare il tuo Dockerfile per selezionare la versione del software AWS IoT Greengrass Core che desideri installare. Puoi anche configurare le variabili di ambiente per scegliere come fornire le risorse durante l'installazione e personalizzare altre opzioni di installazione. Questa sezione descrive come configurare e creare un'immagine AWS IoT Greengrass Docker da un Dockerfile.

Scarica il pacchetto Dockerfile

Puoi scaricare il pacchetto AWS IoT Greengrass Dockerfile da: GitHub

Repository Docker AWS Greengrass

Dopo aver scaricato il pacchetto, estrai il contenuto *`download-directory/`*`aws-greengrass-docker-`*`nucleus-version`* nella cartella sul tuo computer. Il Dockerfile utilizza una versione precedente di Greengrass. È necessario aggiornare il file per utilizzare la versione di Greengrass desiderata.

Specificare la versione del software AWS IoT Greengrass Core

Utilizzate il seguente argomento di compilazione nel Dockerfile per specificare la versione del software AWS IoT Greengrass Core che desiderate utilizzare nell'immagine AWS IoT Greengrass Docker. Per impostazione predefinita, il Dockerfile utilizza l'ultima versione del software Core. AWS IoT Greengrass

`GREENGRASS_RELEASE_VERSION`

La versione del software AWS IoT Greengrass Core. Per impostazione predefinita, il Dockerfile scarica l'ultima versione disponibile del nucleo Greengrass. Imposta il valore sulla versione del nucleo che desideri scaricare.

Impostazione delle variabili di ambiente

Le variabili di ambiente consentono di personalizzare il modo in cui il software AWS IoT Greengrass Core viene installato nel contenitore Docker. Puoi impostare le variabili di ambiente per l'immagine AWS IoT Greengrass Docker in vari modi.

- Per utilizzare le stesse variabili di ambiente per creare più immagini, imposta le variabili di ambiente direttamente nel Dockerfile.
- Se lo usi `docker run` per avviare il contenitore, passa le variabili di ambiente come argomenti nel comando o imposta le variabili di ambiente in un file di variabili di ambiente e poi passa il file come argomento. Per ulteriori informazioni sull'impostazione delle variabili di ambiente in Docker, consulta le variabili di ambiente nella documentazione di Docker.
- Se lo usi `docker-compose up` per avviare il contenitore, imposta le variabili di ambiente in un file di variabili di ambiente e poi passa il file come argomento. Per ulteriori informazioni sull'impostazione delle variabili di ambiente in Compose, consulta la documentazione Docker.

È possibile configurare le seguenti variabili di ambiente per l'immagine AWS IoT Greengrass Docker.

> **Note**
>
> Non modificare la `TINI_KILL_PROCESS_GROUP` variabile nel Dockerfile. Questa variabile consente l'inoltro `SIGTERM` a tutti i PID del gruppo PID in modo che il software AWS IoT Greengrass Core possa spegnersi correttamente quando il contenitore Docker viene arrestato.

`GGC_ROOT_PATH`

(Facoltativo) Il percorso della cartella all'interno del contenitore da utilizzare come root per il software Core. AWS IoT Greengrass

Impostazione predefinita: `/greengrass/v2`

`PROVISION`

(Facoltativo) Determina se il AWS IoT Greengrass Core fornisce AWS le risorse.

- Se si specifica`true`, il software AWS IoT Greengrass Core registra l'immagine del contenitore come AWS IoT oggetto e fornisce le AWS risorse richieste dal dispositivo principale Greengrass. Il software AWS IoT Greengrass Core fornisce qualsiasi AWS IoT oggetto, (facoltativo) un gruppo di AWS IoT oggetti, un ruolo IAM e un alias di AWS IoT ruolo. Per ulteriori informazioni, consulta Esegui AWS IoT Greengrass in un contenitore Docker con provisioning automatico delle risorse.
- Se lo specifichi`false`, devi creare un file di configurazione da fornire al programma di installazione AWS IoT Greengrass Core che specifichi di utilizzare le AWS risorse e i certificati creati manualmente. Per ulteriori informazioni, consulta Esegui AWS IoT Greengrass in un contenitore Docker con provisioning manuale delle risorse.

Default: `false`

`AWS_REGION`

(Facoltativo) Regione AWS Quello che il software AWS IoT Greengrass Core utilizza per recuperare o creare le risorse richieste. AWS

Default: `us-east-1`.

`THING_NAME`

(Facoltativo) Il nome dell'AWS IoTelemento che registri come dispositivo principale. Se l'oggetto con questo nome non esiste nel tuo computerAccount AWS, il software AWS IoT Greengrass Core lo crea.

È necessario specificare se si `PROVISION=true` desidera applicare questo argomento.

Impostazione predefinita: `GreengrassV2IotThing_` più un UUID casuale.

`THING_GROUP_NAME`

(Facoltativo) Il nome del gruppo di oggetti a cui si aggiunge questo dispositivo principale è AWS IoT Se una distribuzione è destinata a questo gruppo di oggetti, questo e gli altri dispositivi principali di quel gruppo ricevono quella distribuzione quando si connette. AWS IoT AWS IoT Greengrass Se il gruppo di oggetti con questo nome non esiste nel tuoAccount AWS, il software AWS IoT Greengrass Core lo crea.

È necessario specificare `PROVISION=true` di applicare questo argomento.

`TES_ROLE_NAME`

(Facoltativo) Il nome del ruolo IAM da utilizzare per acquisire AWS credenziali che consentono al dispositivo principale Greengrass di interagire con AWS i servizi. Se il ruolo con questo nome non esiste nel tuo accountAccount AWS, il software AWS IoT Greengrass Core lo crea con la `GreengrassV2TokenExchangeRoleAccess` policy. Questo ruolo non ha accesso ai tuoi bucket S3 dove ospiti gli artefatti dei componenti. Pertanto, è necessario aggiungere le autorizzazioni ai bucket e agli oggetti S3 degli artefatti quando si crea un componente. Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

Default: `GreengrassV2TokenExchangeRole`

`TES_ROLE_ALIAS_NAME`

(Facoltativo) Il nome dell'alias del AWS IoT ruolo che punta al ruolo IAM che fornisce AWS le credenziali per il dispositivo principale Greengrass. Se l'alias del ruolo con questo nome non esiste nel tuo computerAccount AWS, il software AWS IoT Greengrass Core lo crea e lo indirizza al ruolo IAM da te specificato.

Impostazione predefinita: `GreengrassV2TokenExchangeRoleAlias`

`COMPONENT_DEFAULT_USER`

(Facoltativo) Il nome o l'ID dell'utente e del gruppo di sistema utilizzati dal software AWS IoT Greengrass Core per eseguire i componenti. Specificate l'utente e il gruppo, separati da due punti. Il gruppo è facoltativo. Ad esempio, puoi specificare **`ggc_user:ggc_group`** o **`ggc_user`**.

- Se esegui come root, per impostazione predefinita sono l'utente e il gruppo definiti dal file di configurazione. Se il file di configurazione non definisce un utente e un gruppo, il valore predefinito è. `ggc_user:ggc_group` Se esistono `ggc_user` o `ggc_group` non esistono, il software li crea.
- Se si esegue come utente non root, il software AWS IoT Greengrass Core utilizza quell'utente per eseguire i componenti.
- Se non specificate un gruppo, il software AWS IoT Greengrass Core utilizza il gruppo principale dell'utente del sistema.

Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti.

`DEPLOY_DEV_TOOLS`

Definisce se scaricare e distribuire il componente Greengrass CLI nell'immagine del contenitore. È possibile utilizzare la Greengrass CLI per sviluppare ed eseguire il debug dei componenti localmente.

> **Important**
>
> Si consiglia di utilizzare questo componente solo in ambienti di sviluppo, non in ambienti di produzione. Questo componente fornisce l'accesso a informazioni e operazioni che in genere non sono necessarie in un ambiente di produzione. Segui il principio del privilegio minimo distribuendo questo componente solo sui dispositivi principali dove ne hai bisogno.

Impostazione predefinita: `false`

`INIT_CONFIG`

(Facoltativo) Il percorso del file di configurazione da utilizzare per installare il software AWS IoT Greengrass Core. È possibile utilizzare questa opzione per configurare nuovi dispositivi core Greengrass con una configurazione del nucleo specifica o per specificare risorse assegnate manualmente, ad esempio. È necessario montare il file di configurazione nel percorso specificato in questo argomento.

`TRUSTED_PLUGIN`

Questa funzionalità è disponibile per la versione 2.4.0 e successive del componente Greengrass nucleus.

(Facoltativo) Il percorso di un file JAR da caricare come plugin affidabile. Utilizzate questa opzione per fornire i file JAR del plug-in di provisioning, ad esempio da installare con fleet provisioning o custom provisioning.

`THING_POLICY_NAME`

Questa funzionalità è disponibile per la versione 2.4.0 e successive del componente Greengrass nucleus.

(Facoltativo) Il nome della AWS IoT policy da allegare al thing certificate di questo dispositivo principale. AWS IoT Se la AWS IoT politica con questo nome non esiste nel tuo computer, Account AWS il software AWS IoT Greengrass Core la crea.

È necessario specificare `PROVISION=true` se applicare questo argomento.

> **Note**
>
> Il software AWS IoT Greengrass Core crea una AWS IoT politica permissiva per impostazione predefinita. È possibile definire l'ambito di questa politica o creare una politica personalizzata in cui limitare le autorizzazioni per il proprio caso d'uso. Per ulteriori informazioni, consulta AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali.

Specificate le dipendenze da installare

L'istruzione RUN nel AWS IoT Greengrass Dockerfile prepara l'ambiente contenitore per eseguire il programma di installazione del AWS IoT Greengrass software Core. È possibile personalizzare le dipendenze installate prima dell'esecuzione del programma di installazione del software AWS IoT Greengrass Core nel contenitore Docker.

Costruisci l'immagine AWS IoT Greengrass

Usa il AWS IoT Greengrass Dockerfile per creare un'immagine del AWS IoT Greengrass contenitore. Puoi utilizzare la CLI Docker o la CLI Docker Compose per creare l'immagine e avviare il contenitore. Puoi anche utilizzare la CLI Docker per creare l'immagine e quindi utilizzare Docker Compose per avviare il contenitore da quell'immagine.

Docker

1. Sulla macchina host, esegui il seguente comando per passare alla directory che contiene il Dockerfile configurato.

   ```
   cd download-directory/aws-greengrass-docker-nucleus-version
   ```

2. Esegui il comando seguente per creare l'immagine del AWS IoT Greengrass contenitore dal Dockerfile.

   ```
   sudo docker build -t "platform/aws-iot-greengrass:nucleus-version" ./
   ```

Docker Compose

1. Sul computer host, esegui il comando seguente per passare alla directory che contiene il Dockerfile e il file Compose.

   ```
   cd download-directory/aws-greengrass-docker-nucleus-version
   ```

2. Esegui il comando seguente per utilizzare il file Compose per creare l'immagine del contenitore. AWS IoT Greengrass

   ```
   docker-compose -f docker-compose.yml build
   ```

L'immagine del AWS IoT Greengrass contenitore è stata creata con successo. Nell'immagine Docker è installato il software AWS IoT Greengrass Core. Ora puoi eseguire il software AWS IoT Greengrass Core in un contenitore Docker.

Esegui AWS IoT Greengrass in un contenitore Docker con provisioning automatico delle risorse

Questo tutorial mostra come installare ed eseguire il software AWS IoT Greengrass Core in un contenitore Docker con AWS risorse assegnate automaticamente e strumenti di sviluppo locali. Puoi utilizzare questo ambiente di sviluppo per esplorare le AWS IoT Greengrass funzionalità di un contenitore Docker. Il software richiede AWS credenziali per fornire queste risorse e distribuire gli strumenti di sviluppo locali.

Se non è possibile fornire AWS le credenziali al contenitore, è possibile fornire le AWS risorse necessarie al funzionamento del dispositivo principale. Puoi anche distribuire gli strumenti di sviluppo su un dispositivo principale da utilizzare come dispositivo di sviluppo. Ciò consente di fornire meno autorizzazioni al dispositivo quando si esegue il contenitore. Per ulteriori informazioni, consulta Esegui AWS IoT Greengrass in un contenitore Docker con provisioning manuale delle risorse.

Prerequisiti

Per completare questo tutorial, è necessario quanto segue.

- Un Account AWS. Se non lo hai, consultare Configura un Account AWS.
- Un utente AWS IAM con le autorizzazioni per effettuare il provisioning delle risorse AWS IoT e IAM per un dispositivo core Greengrass. Il programma di installazione del software AWS IoT

Greengrass Core utilizza AWS le tue credenziali per effettuare automaticamente il provisioning di queste risorse. Per informazioni sulla politica minima di IAM per il provisioning automatico delle risorse, consulta. Policy IAM minima per l'installatore per il provisioning delle risorse

- Un'immagine AWS IoT Greengrass Docker. Puoi creare un'immagine dal AWS IoT Greengrass Dockerfile.
- Il computer host su cui si esegue il contenitore Docker deve soddisfare i seguenti requisiti:
  - Un sistema operativo basato su Linux con una connessione Internet.
  - Docker Engine versione 18.09 o successiva.
  - (Facoltativo) Docker Compose versione 1.22 o successiva. Docker Compose è necessario solo se si desidera utilizzare la CLI Docker Compose per eseguire le immagini Docker.

Configurazione delle credenziali AWS

In questo passaggio, crei un file di credenziali sul computer host che contiene le tue credenziali di sicurezza. AWS Quando esegui l'immagine AWS IoT Greengrass Docker, devi montare la cartella che contiene questo file di credenziali `/root/.aws/` nel contenitore Docker. Il AWS IoT Greengrass programma di installazione utilizza queste credenziali per fornire risorse nel tuo. Account AWS Per informazioni sulla politica minima di IAM richiesta dal programma di installazione per il provisioning automatico delle risorse, consulta. Policy IAM minima per l'installatore per il provisioning delle risorse

1. Recuperate uno dei seguenti elementi.
   - Credenziali a lungo termine per un utente IAM. Per informazioni su come recuperare le credenziali a lungo termine, consulta Managing access keys for IAM users nella IAM User Guide.
   - (Consigliato) Credenziali temporanee per un ruolo IAM. Per informazioni su come recuperare le credenziali temporanee, consulta Using temporary security credentials with the AWS CLI nella IAM User Guide.
2. Crea una cartella in cui inserire il file delle credenziali.

   ```
   mkdir ./greengrass-v2-credentials
   ```

3. Utilizzate un editor di testo per creare un file di configurazione denominato `credentials` nella `./greengrass-v2-credentials` cartella.

Ad esempio, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il `credentials` file.

```
nano ./greengrass-v2-credentials/credentials
```

4. Aggiungi AWS le tue credenziali al `credentials` file nel seguente formato.

```
[default]
aws_access_key_id = AKIAIOSFODNN7EXAMPLE
aws_secret_access_key = wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
aws_session_token
 = AQoEXAMPLEH4aoAH0gNCAPy...truncated...zrkuWJOgQs8IZZaIv2BXIa2R4Olgk
```

Includi `aws_session_token` solo per le credenziali temporanee.

> **Important**
>
> Rimuovi il file delle credenziali dal computer host dopo aver avviato il AWS IoT Greengrass contenitore. Se non rimuovi il file delle credenziali, le tue AWS credenziali rimarranno montate all'interno del contenitore. Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core in un contenitore.

Crea un file di ambiente

Questo tutorial utilizza un file di ambiente per impostare le variabili di ambiente che verranno passate al programma di installazione del software AWS IoT Greengrass Core all'interno del contenitore Docker. Puoi anche usare l'`--env`argomento `-e` or nel `docker run` comando per impostare le variabili di ambiente nel contenitore Docker oppure puoi impostare le variabili in un `environment` blocco del file. `docker-compose.yml`

1. Usa un editor di testo per creare un file di ambiente denominato`.env`.

   Ad esempio, su un sistema basato su Linux, puoi eseguire il seguente comando per usare GNU nano per crearlo `.env` nella directory corrente.

```
nano .env
```

2. Copiate il seguente contenuto nel file.

```
GGC_ROOT_PATH=/greengrass/v2
AWS_REGION=region
PROVISION=true
THING_NAME=MyGreengrassCore
THING_GROUP_NAME=MyGreengrassCoreGroup
TES_ROLE_NAME=GreengrassV2TokenExchangeRole
TES_ROLE_ALIAS_NAME=GreengrassCoreTokenExchangeRoleAlias
COMPONENT_DEFAULT_USER=ggc_user:ggc_group
```

Quindi, sostituisci i seguenti valori.

- */greengrass/v2*. La cartella principale di Greengrass che si desidera utilizzare per l'installazione. Utilizzate la variabile di `GGC_ROOT` ambiente per impostare questo valore.
- *regione*. Il Regione AWS luogo in cui hai creato le risorse.
- *MyGreengrassCore*. Nome del nuovo oggetto AWS IoT. Se l'oggetto non esiste, il programma di installazione lo crea. Il programma di installazione scarica i certificati per autenticarsi come oggetto. AWS IoT
- *MyGreengrassCoreGroup*. Il nome del gruppo di AWS IoT cose. Se il gruppo di oggetti non esiste, il programma di installazione lo crea e vi aggiunge l'oggetto. Se il gruppo di oggetti esiste e dispone di una distribuzione attiva, il dispositivo principale scarica ed esegue il software specificato dalla distribuzione.
- *GreenGrass TokenExchangeRole* V2. Sostituisci con il nome del ruolo di scambio di token IAM che consente al dispositivo principale Greengrass di ottenere credenziali temporaneeAWS. *Se il ruolo non esiste, il programma di installazione lo crea e crea e allega una policy denominata GreengrassV2 Access. TokenExchangeRole* Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.
- *GreengrassCoreTokenExchangeRoleAlias*. L'alias del ruolo di scambio di token. Se l'alias del ruolo non esiste, il programma di installazione lo crea e lo indirizza al ruolo di scambio di token IAM specificato. Per ulteriori informazioni, consulta la pagina

> **Note**
>
> È possibile impostare la variabile di `DEPLOY_DEV_TOOLS` ambiente su `true` per distribuire il componente Greengrass CLI, che consente di sviluppare componenti

personalizzati all'interno del contenitore Docker. Si consiglia di utilizzare questo componente solo in ambienti di sviluppo, non in ambienti di produzione. Questo componente fornisce l'accesso a informazioni e operazioni che in genere non sono necessarie in un ambiente di produzione. Segui il principio del privilegio minimo distribuendo questo componente solo sui dispositivi principali dove ne hai bisogno.

Esegui il software AWS IoT Greengrass Core in un contenitore

Questo tutorial mostra come avviare l'immagine Docker che hai creato in un contenitore Docker. Puoi utilizzare la CLI Docker o la CLI Docker Compose per AWS IoT Greengrass eseguire l'immagine del software Core in un contenitore Docker.

Docker

1. Esegui il comando seguente per avviare il contenitore Docker.

```
docker run --rm --init -it --name docker-image \
 -v path/to/greengrass-v2-credentials:/root/.aws/:ro \
 --env-file .env \
 -p 8883 \
 your-container-image:version
```

Questo comando di esempio utilizza i seguenti argomenti per docker run:

- `--rm`. Pulisce il contenitore quando esce.
- `--init`. Utilizza un processo di inizializzazione nel contenitore.

> **Note**
>
> L'`--init`argomento è necessario per chiudere il software AWS IoT Greengrass Core quando si arresta il contenitore Docker.

- `-it`. (Facoltativo) Esegue il contenitore Docker in primo piano come processo interattivo. Puoi sostituirlo con l'`-d`argomento per eseguire invece il contenitore Docker in modalità distaccata. Per ulteriori informazioni, consulta Detached vs foreground nella documentazione Docker.
- `--name`. Esegue un contenitore denominato `aws-iot-greengrass`

- `-v`. Monta un volume nel contenitore Docker per rendere il file di configurazione e i file di certificato disponibili per l'AWS IoT Greengrassesecuzione all'interno del contenitore.
- `--env-file`. (Facoltativo) Specifica il file di ambiente per impostare le variabili di ambiente che verranno passate al programma di installazione del software AWS IoT Greengrass Core all'interno del contenitore Docker. Questo argomento è richiesto solo se è stato creato un file di ambiente per impostare le variabili di ambiente. Se non hai creato un file di ambiente, puoi utilizzare `--env` gli argomenti per impostare le variabili di ambiente direttamente nel comando Docker run.
- `-p`. (Facoltativo) Pubblica la porta container 8883 sulla macchina host. Questo argomento è necessario se si desidera connettersi e comunicare tramite MQTT perché AWS IoT Greengrass utilizza la porta 8883 per il traffico MQTT. Per aprire altre porte, utilizzate argomenti aggiuntivi. `-p`

> **Note**
>
> Per eseguire il contenitore Docker con maggiore sicurezza, puoi utilizzare gli `--cap-add` argomenti `--cap-drop` e per abilitare selettivamente le funzionalità Linux per il tuo contenitore. Per ulteriori informazioni, consulta Privilegi di runtime e funzionalità Linux nella documentazione Docker.

2. Rimuovi le credenziali dal dispositivo `./greengrass-v2-credentials` host.

```
rm -rf ./greengrass-v2-credentials
```

> **Important**
>
> Stai rimuovendo queste credenziali perché forniscono ampie autorizzazioni di cui il dispositivo principale ha bisogno solo durante la configurazione. Se non rimuovi queste credenziali, i componenti Greengrass e gli altri processi in esecuzione nel contenitore possono accedervi. Se devi fornire AWS credenziali a un componente Greengrass, utilizza il servizio di scambio di token. Per ulteriori informazioni, consulta Interagisci con AWS i servizi.

Docker Compose

1. Usa un editor di testo per creare un file Docker Compose denominato. `docker-compose.yml`

   Ad esempio, su un sistema basato su Linux, puoi eseguire il seguente comando per usare GNU nano per crearlo nella directory corrente. `docker-compose.yml`

   ```
   nano docker-compose.yml
   ```

   > **Note**
   >
   > È inoltre possibile scaricare e utilizzare la versione più recente del file Compose fornito AWS da. GitHub

2. Aggiungi il seguente contenuto al file Compose. Il file si presenta in maniera simile al seguente frammento: Sostituisci *`docker-image`* con il nome della tua immagine Docker.

   ```
   version: '3.7'

   services:
     greengrass:
       init: true
       container_name: aws-iot-greengrass
       image: docker-image
       volumes:
         - ./greengrass-v2-credentials:/root/.aws/:ro
       env_file: .env
       ports:
         - "8883:8883"
   ```

   I seguenti parametri in questo file Compose di esempio sono facoltativi:

   - `ports`—Pubblica le porte del contenitore 8883 sul computer host. Questo parametro è necessario se si desidera connettersi e comunicare tramite MQTT perché AWS IoT Greengrass utilizza la porta 8883 per il traffico MQTT.
   - `env_file`: specifica il file di ambiente per impostare le variabili di ambiente che verranno passate al programma di installazione del software AWS IoT Greengrass Core all'interno del contenitore Docker. Questo parametro è richiesto solo se è stato creato un file di

ambiente per impostare le variabili di ambiente. Se non avete creato un file di ambiente, potete utilizzare il parametro environment per impostare le variabili direttamente nel file Compose.

> **Note**
>
> Per eseguire il contenitore Docker con maggiore sicurezza, puoi utilizzare `cap_drop` e `cap_add` nel file Compose per abilitare selettivamente le funzionalità Linux per il tuo contenitore. Per ulteriori informazioni, consulta Privilegi di runtime e funzionalità Linux nella documentazione Docker.

3. Esegui il comando seguente per avviare il contenitore Docker.

```
docker-compose -f docker-compose.yml up
```

4. Rimuovi le credenziali dal `./greengrass-v2-credentials` dispositivo host.

```
rm -rf ./greengrass-v2-credentials
```

> **Important**
>
> Stai rimuovendo queste credenziali perché forniscono ampie autorizzazioni di cui il dispositivo principale ha bisogno solo durante la configurazione. Se non rimuovi queste credenziali, i componenti Greengrass e gli altri processi in esecuzione nel contenitore possono accedervi. Se devi fornire AWS credenziali a un componente Greengrass, utilizza il servizio di scambio di token. Per ulteriori informazioni, consulta Interagisci con AWS i servizi.

Passaggi successivi

AWS IoT GreengrassIl software principale è ora in esecuzione in un contenitore Docker. Esegui il comando seguente per recuperare l'ID del contenitore attualmente in esecuzione.

```
docker ps
```

È quindi possibile eseguire il comando seguente per accedere al contenitore ed esplorare il software AWS IoT Greengrass Core in esecuzione all'interno del contenitore.

```
docker exec -it container-id /bin/bash
```

Per informazioni sulla creazione di un componente semplice, consulta Fase 4: Sviluppa e testa un componente sul tuo dispositivo in Tutorial: Nozioni di base su AWS IoT Greengrass V2

> **Note**
>
> Quando esegui comandi all'interno del contenitore Docker, tali comandi non vengono registrati nei log Docker. `docker exec` Per registrare i comandi nei log Docker, collega una shell interattiva al contenitore Docker. Per ulteriori informazioni, consulta Collega una shell interattiva al contenitore Docker.

Il file di registro AWS IoT Greengrass Core viene chiamato `greengrass.log` e si trova in. */greengrass/v2*`/logs` Nella stessa directory si trovano anche i file di registro dei componenti. Per copiare i log di Greengrass in una directory temporanea sull'host, esegui il seguente comando:

```
docker cp container-id:/greengrass/v2/logs /tmp/logs
```

Se desideri mantenere i log dopo l'uscita o la rimozione di un contenitore, ti consigliamo di collegare solo la */greengrass/v2*`/logs` directory alla directory dei log temporanei sull'host invece di montare l'intera directory Greengrass. Per ulteriori informazioni, consulta Mantieni i log Greengrass all'esterno del contenitore Docker.

Per fermare un contenitore Docker in esecuzione, esegui o. AWS IoT Greengrass `docker stop` `docker-compose -f docker-compose.yml stop` Questa azione invia `SIGTERM` al processo Greengrass e chiude tutti i processi associati che sono stati avviati nel contenitore. Il contenitore Docker viene inizializzato con l'`docker-init`eseguibile come processo PID 1, il che aiuta a rimuovere eventuali processi zombie rimanenti. Per ulteriori informazioni, consulta Specificare un processo di inizializzazione nella documentazione di Docker.

Per informazioni sulla risoluzione dei problemi relativi all'esecuzione AWS IoT Greengrass in un contenitore Docker, consulta. Risoluzione dei problemi di AWS IoT Greengrass in un container Docker

Esegui AWS IoT Greengrass in un contenitore Docker con provisioning manuale delle risorse

Questo tutorial mostra come installare ed eseguire il software AWS IoT Greengrass Core in un contenitore Docker con risorse assegnate manualmente. AWS

Argomenti

Prerequisiti

Per completare questo tutorial, è necessario quanto segue:

- Un Account AWS. Se non lo hai, consultare Configura un Account AWS.
- Un'immagine AWS IoT Greengrass Docker. Puoi creare un'immagine dal AWS IoT Greengrass Dockerfile.
- Il computer host su cui si esegue il contenitore Docker deve soddisfare i seguenti requisiti:
  - Un sistema operativo basato su Linux con una connessione Internet.
  - Docker Engine versione 18.09 o successiva.
  - (Facoltativo) Docker Compose versione 1.22 o successiva. Docker Compose è necessario solo se si desidera utilizzare la CLI Docker Compose per eseguire le immagini Docker.

AWS IoTRecupera gli endpoint

Ottieni gli AWS IoT endpoint per te e salvali per Account AWS utilizzarli in un secondo momento. Il tuo dispositivo utilizza questi endpoint per connettersi a. AWS IoT Esegui questa operazione:

1. Ottieni l'endpoint di AWS IoT dati per il tuo. Account AWS

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS
   ```

   La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

   ```
   {
     "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Ottieni l'endpoint delle AWS IoT credenziali per il tuo. Account AWS

   ```
   aws iot describe-endpoint --endpoint-type iot:CredentialProvider
   ```

   La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

   ```
   {
     "endpointAddress": "device-credentials-prefix.credentials.iot.us-
   west-2.amazonaws.com"
   }
   ```

Crea qualsiasi cosa AWS IoT

AWS IoTle cose rappresentano dispositivi ed entità logiche a cui si connettonoAWS IoT. I dispositivi core Greengrass sono AWS IoT cose. Quando registri un dispositivo come AWS IoT oggetto, quel dispositivo può utilizzare un certificato digitale con cui autenticarsi. AWS

In questa sezione, crei AWS IoT qualcosa che rappresenta il tuo dispositivo.

Per creare qualsiasi AWS IoT cosa

1. Crea AWS IoT qualcosa per il tuo dispositivo. Sul tuo computer di sviluppo, esegui il seguente comando.

- Sostituisci *MyGreengrassCore*con il nome dell'oggetto da usare. Questo nome è anche il nome del dispositivo principale Greengrass.

  > **Note**
  >
  > Il nome dell'oggetto non può contenere i due punti (`:`).

```
aws iot create-thing --thing-name MyGreengrassCore
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "thingName": "MyGreengrassCore",
  "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "thingId": "8cb4b6cd-268e-495d-b5b9-1713d71dbf42"
}
```

2. (Facoltativo) Aggiungere l'AWS IoToggetto a un gruppo di oggetti nuovo o esistente. Utilizzi i gruppi di cose per gestire flotte di dispositivi core Greengrass. Quando distribuisci componenti software sui tuoi dispositivi, puoi scegliere come target singoli dispositivi o gruppi di dispositivi. È possibile aggiungere un dispositivo a un gruppo di cose con una distribuzione Greengrass attiva per distribuire i componenti software di quel gruppo di cose sul dispositivo. Esegui questa operazione:

   a. (Facoltativo) Crea un gruppo di AWS IoT cose.

      - Sostituire *MyGreengrassCoreGroup*con il nome del gruppo di oggetti da creare.

        > **Note**
        >
        > Il nome del gruppo di cose non può contenere i due punti (`:`).

```
aws iot create-thing-group --thing-group-name MyGreengrassCoreGroup
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "thingGroupName": "MyGreengrassCoreGroup",
  "thingGroupArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
MyGreengrassCoreGroup",
  "thingGroupId": "4df721e1-ff9f-4f97-92dd-02db4e3f03aa"
}
```

b. Aggiungere l'AWS IoToggetto a un gruppo di oggetti.

- Sostituiscilo *MyGreengrassCore*con il nome del tuo AWS IoT oggetto.
- Sostituisci *MyGreengrassCoreGroup*con il nome del gruppo di oggetti.

```
aws iot add-thing-to-thing-group --thing-name MyGreengrassCore --thing-group-
name MyGreengrassCoreGroup
```

Il comando non produce alcun output se la richiesta ha esito positivo.

Crea il certificato dell'oggetto

Quando si registra un dispositivo come AWS IoT oggetto, quel dispositivo può utilizzare un certificato digitale con AWS cui autenticarsi. Questo certificato consente al dispositivo di comunicare con AWS IoT eAWS IoT Greengrass.

In questa sezione, crei e scarichi certificati a cui il tuo dispositivo può connettersiAWS.

Per creare il certificato dell'oggetto

1. Crea una cartella in cui scaricare i certificati relativi all'AWS IoToggetto.

```
mkdir greengrass-v2-certs
```

2. Crea e scarica i certificati relativi all'AWS IoToggetto.

```
aws iot create-keys-and-certificate --set-as-active --certificate-pem-outfile
 greengrass-v2-certs/device.pem.crt --public-key-outfile greengrass-v2-certs/
public.pem.key --private-key-outfile greengrass-v2-certs/private.pem.key
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "certificateArn": "arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
  "certificateId":
 "aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4",
  "certificatePem": "-----BEGIN CERTIFICATE-----
MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w
 0BAQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZ
 WF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIw
 EAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5
 jb20wHhcNMTEwNDI1MjA0NTIxWhcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
 MCVVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
 WF6b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
 HzAdBgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
 BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
 k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
 ITxOUSQv7c7ugFFDzQGBzZswY6786m86gpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
 AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4nUhVVxYUntneD9+h8Mg9q6q+auN
 KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FkbFFBjvSfpJIlJ00zbhNYS5f6Guo
 EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTbNYiytVbZPQUQ5Yaxu2jXnimvw
 3rrszlaEXAMPLE=
-----END CERTIFICATE-----",
  "keyPair": {
    "PublicKey": "-----BEGIN PUBLIC KEY-----\
MIIBIjANBgkqhkEXAMPLEQEFAAOCAQ8AMIIBCgKCAQEAEXAMPLE1nnyJwKSMHw4h\
MMEXAMPLEuuN/dMAS3fyce8DW/4+EXAMPLEyjmoF/YVF/gHr99VEEXAMPLE5VF13\
59VK7cEXAMPLE67GK+y+jikqXOgHh/xJTwo
+sGpWEXAMPLEDz18xOd2ka4tCzuWEXAMPLEahJbYkCPUBSU8opVkR7qkEXAMPLE1DR6sx2HocliOOLtu6Fkw91swQWE
\\GB3ZPrNh0PzQYvjUStZeccyNCx2EXAMPLEvp9mQOUXP6plfgxwKRX2fEXAMPLEDa\
hJLXkX3rHU2xbxJSq7D+XEXAMPLEcw+LyFhI5mgFRl88eGdsAEXAMPLElnI9EesG\
FQIDAQAB\
-----END PUBLIC KEY-----\
",
    "PrivateKey": "-----BEGIN RSA PRIVATE KEY-----\
key omitted for security reasons\
-----END RSA PRIVATE KEY-----\
"
  }
}
```

Salva l'Amazon Resource Name (ARN) del certificato da utilizzare per configurare il certificato in un secondo momento.

Configura il certificato del oggetto

Allega il thing certificate all'AWS IoToggetto che hai creato in precedenza e aggiungi una AWS IoT policy al certificato per definire le AWS IoT autorizzazioni per il dispositivo principale.

Per configurare il certificato dell'oggetto

1. Allega il certificato all'AWS IoToggetto.

   - Sostituiscilo *MyGreengrassCore*con il nome del tuo AWS IoT oggetto.
   - Sostituisci il certificato Amazon Resource Name (ARN) con l'ARN del certificato che hai creato nel passaggio precedente.

   ```
   aws iot attach-thing-principal --thing-name MyGreengrassCore
    --principal arn:aws:iot:us-west-2:123456789012:cert/
   aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
   ```

   Il comando non produce alcun output se la richiesta ha esito positivo.

2. Crea e allega una AWS IoT policy che definisca le AWS IoT autorizzazioni per il tuo dispositivo principale Greengrass. La seguente politica consente l'accesso a tutti gli argomenti MQTT e alle operazioni di Greengrass, in modo che il dispositivo funzioni con applicazioni personalizzate e modifiche future che richiedono nuove operazioni Greengrass. È possibile limitare questa politica in base al proprio caso d'uso. Per ulteriori informazioni, consulta AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali.

   Se hai già configurato un dispositivo Greengrass core, puoi allegare la AWS IoT relativa policy invece di crearne una nuova.

   Esegui questa operazione:

   a. Crea un file che contenga il documento di AWS IoT policy richiesto dai dispositivi core Greengrass.

      Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

      ```
      nano greengrass-v2-iot-policy.json
      ```

Copiate il seguente codice JSON nel file.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "iot:Connect",
        "greengrass:*"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

b. Crea una AWS IoT politica dal documento di policy.

- Sostituisci *GreenGrassV2IoT ThingPolicy* con il nome della policy da creare.

```
aws iot create-policy --policy-name GreengrassV2IoTThingPolicy --policy-
document file://greengrass-v2-iot-policy.json
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "policyName": "GreengrassV2IoTThingPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
  "policyDocument": "{
    \\"Version\\": \\"2012-10-17\\",
    \\"Statement\\": [
      {
        \\"Effect\\": \\"Allow\\",
        \\"Action\\": [
          \\"iot:Publish\\",
```

```
          \\"iot:Subscribe\\",
          \\"iot:Receive\\",
          \\"iot:Connect\\",
          \\"greengrass:*\\"
        ],
        \\"Resource\\": [
          \\"*\\"
        ]
      }
    ]
  }",
  "policyVersionId": "1"
}
```

c. Allega la AWS IoT policy al certificato dell'AWS IoToggetto.

- Sostituisci *GreenGrassV2IoT ThingPolicy* con il nome della policy da allegare.
- Sostituisci l'ARN di destinazione con l'ARN del certificato per il tuo oggetto. AWS IoT

```
aws iot attach-policy --policy-name GreengrassV2IoTThingPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

Il comando non ha alcun output se la richiesta ha esito positivo.

Crea un ruolo di scambio di token

I dispositivi core Greengrass utilizzano un ruolo di servizio IAM, chiamato token exchange role, per autorizzare le chiamate ai servizi. AWS Il dispositivo utilizza il provider di AWS IoT credenziali per ottenere AWS credenziali temporanee per questo ruolo, che consente al dispositivo di interagireAWS IoT, inviare log ad Amazon CloudWatch Logs e scaricare elementi dei componenti personalizzati da Amazon S3. Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

Si utilizza un alias di AWS IoT ruolo per configurare il ruolo di scambio di token per i dispositivi principali Greengrass. Gli alias di ruolo consentono di modificare il ruolo di scambio di token per un dispositivo ma mantengono invariata la configurazione del dispositivo. Per ulteriori informazioni, consulta Autorizzazione delle chiamate dirette ai AWS servizi nella Guida per gli AWS IoT Coresviluppatori.

In questa sezione, crei un ruolo IAM per lo scambio di token e un alias di AWS IoT ruolo che rimanda al ruolo. Se hai già configurato un dispositivo principale Greengrass, puoi utilizzare il ruolo di scambio di token e l'alias del ruolo invece di crearne di nuovi. Quindi, configuri il dispositivo in modo che utilizzi quel ruolo e quell'alias. AWS IoT

Per creare un ruolo IAM per lo scambio di token

1. Crea un ruolo IAM che il tuo dispositivo possa utilizzare come ruolo di scambio di token. Esegui questa operazione:

   a. Crea un file che contenga il documento sulla politica di fiducia richiesto dal ruolo di scambio di token.

      Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

      ```
      nano device-role-trust-policy.json
      ```

      Copia il seguente codice JSON nel file.

      ```
      {
        "Version": "2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Principal": {
              "Service": "credentials.iot.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
          }
        ]
      }
      ```

   b. Crea il ruolo di scambio di token con il documento sulla politica di fiducia.

      - Sostituisci *greengrassV2 TokenExchangeRole* con il nome del ruolo IAM da creare.

      ```
      aws iam create-role --role-name GreengrassV2TokenExchangeRole --assume-role-
      policy-document file://device-role-trust-policy.json
      ```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "Role": {
    "Path": "/",
    "RoleName": "GreengrassV2TokenExchangeRole",
    "RoleId": "AROAZ2YMUHYHK5OKM77FB",
    "Arn": "arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole",
    "CreateDate": "2021-02-06T00:13:29+00:00",
    "AssumeRolePolicyDocument": {
      "Version": "2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Principal": {
            "Service": "credentials.iot.amazonaws.com"
          },
          "Action": "sts:AssumeRole"
        }
      ]
    }
  }
```

c. Crea un file che contenga il documento sulla politica di accesso richiesto dal ruolo di scambio di token.

Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

```
nano device-role-access-policy.json
```

Copia il seguente codice JSON nel file.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
```

```
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> Questa politica di accesso non consente l'accesso agli artefatti dei componenti nei bucket S3. Per distribuire componenti personalizzati che definiscono gli artefatti in Amazon S3, devi aggiungere autorizzazioni al ruolo per consentire al dispositivo principale di recuperare gli artefatti dei componenti. Per ulteriori informazioni, consulta Consenti l'accesso ai bucket S3 per gli artefatti dei componenti.
> Se non disponi ancora di un bucket S3 per gli artefatti dei componenti, puoi aggiungere queste autorizzazioni in un secondo momento dopo aver creato un bucket.

d. Crea la policy IAM dal documento di policy.

- Sostituisci *GreenGrassV2 TokenExchangeRoleAccess* con il nome della policy IAM da creare.

```
aws iam create-policy --policy-name GreengrassV2TokenExchangeRoleAccess --
policy-document file://device-role-access-policy.json
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "Policy": {
    "PolicyName": "GreengrassV2TokenExchangeRoleAccess",
    "PolicyId": "ANPAZ2YMUHYHACI7C5Z66",
    "Arn": "arn:aws:iam::123456789012:policy/
GreengrassV2TokenExchangeRoleAccess",
    "Path": "/",
    "DefaultVersionId": "v1",
    "AttachmentCount": 0,
```

```
    "PermissionsBoundaryUsageCount": 0,
    "IsAttachable": true,
    "CreateDate": "2021-02-06T00:37:17+00:00",
    "UpdateDate": "2021-02-06T00:37:17+00:00"
  }
}
```

e. Allega la policy IAM al ruolo di scambio di token.

- Sostituisci *greengrassV2 TokenExchangeRole* con il nome del ruolo IAM.
- Sostituisci l'ARN della policy con l'ARN della policy IAM che hai creato nel passaggio precedente.

```
aws iam attach-role-policy --role-name GreengrassV2TokenExchangeRole --policy-
arn arn:aws:iam::123456789012:policy/GreengrassV2TokenExchangeRoleAccess
```

Il comando non ha alcun output se la richiesta ha esito positivo.

2. Crea un alias di AWS IoT ruolo che punti al ruolo di scambio di token.

- Sostituiscilo *GreengrassCoreTokenExchangeRoleAlias*con il nome dell'alias del ruolo da creare.
- Sostituisci il ruolo ARN con l'ARN del ruolo IAM creato nel passaggio precedente.

```
aws iot create-role-alias --role-alias GreengrassCoreTokenExchangeRoleAlias --role-
arn arn:aws:iam::123456789012:role/GreengrassV2TokenExchangeRole
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "roleAlias": "GreengrassCoreTokenExchangeRoleAlias",
  "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias"
}
```

> **Note**
>
> Per creare un alias di ruolo, devi disporre dell'autorizzazione a passare il ruolo IAM per lo scambio di token a. AWS IoT Se ricevi un messaggio di errore quando tenti di creare un alias di ruolo, verifica che AWS l'utente disponga di questa autorizzazione. Per ulteriori informazioni, consulta Concessione a un utente delle autorizzazioni per il trasferimento di un ruolo a un AWS servizio nella Guida per l'AWS Identity and Access Managementutente.

3. Crea e allega una AWS IoT policy che consenta al tuo dispositivo principale Greengrass di utilizzare l'alias del ruolo per assumere il ruolo di scambio di token. Se hai già configurato un dispositivo principale Greengrass, puoi allegare la sua AWS IoT politica di alias di ruolo invece di crearne uno nuovo. Esegui questa operazione:

   a. (Facoltativo) Create un file che contenga il documento di AWS IoT policy richiesto dall'alias di ruolo.

      Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

      ```
      nano greengrass-v2-iot-role-alias-policy.json
      ```

      Copia il seguente codice JSON nel file.

      - Sostituisci l'ARN della risorsa con l'ARN del tuo alias di ruolo.

      ```
      {
        "Version":"2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": "iot:AssumeRoleWithCertificate",
            "Resource": "arn:aws:iot:us-west-2:123456789012:rolealias/
      GreengrassCoreTokenExchangeRoleAlias"
          }
        ]
      }
      ```

b. Crea una AWS IoT politica dal documento di policy.

- Sostituisci *GreengrassCoreTokenExchangeRoleAliasPolicy*con il nome della AWS IoT politica da creare.

```
aws iot create-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --policy-document file://greengrass-v2-iot-role-alias-policy.json
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "policyName": "GreengrassCoreTokenExchangeRoleAliasPolicy",
  "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassCoreTokenExchangeRoleAliasPolicy",
  "policyDocument": "{
    \\"Version\\":\\"2012-10-17\\",
    \\"Statement\\": [
      {
        \\"Effect\\": \\"Allow\\",
        \\"Action\\": \\"iot:AssumeRoleWithCertificate\\",
        \\"Resource\\": \\"arn:aws:iot:us-west-2:123456789012:rolealias/
GreengrassCoreTokenExchangeRoleAlias\\"
      }
    ]
  }",
  "policyVersionId": "1"
}
```

c. Allega la AWS IoT policy al certificato dell'AWS IoToggetto.

- Sostituisci *GreengrassCoreTokenExchangeRoleAliasPolicy*con il nome della AWS IoT politica relativa agli alias del ruolo.
- Sostituisci l'ARN di destinazione con l'ARN del certificato per il tuo oggetto. AWS IoT

```
aws iot attach-policy --policy-name GreengrassCoreTokenExchangeRoleAliasPolicy
 --target arn:aws:iot:us-west-2:123456789012:cert/
aa0b7958770878eabe251d8a7ddd547f4889c524c9b574ab9fbf65f32248b1d4
```

Il comando non ha alcun output se la richiesta ha esito positivo.

Scarica i certificati sul dispositivo

In precedenza, avevi scaricato il certificato del dispositivo sul computer di sviluppo. In questa sezione, scarichi il certificato Amazon Root Certificate Authority (CA). Quindi, se prevedi di eseguire il software AWS IoT Greengrass Core in Docker su un computer diverso da quello di sviluppo, copi i certificati su quel computer host. Il software AWS IoT Greengrass Core utilizza questi certificati per connettersi al servizio AWS IoT cloud.

Per scaricare i certificati sul dispositivo

1. Sul tuo computer di sviluppo, scarica il certificato Amazon Root Certificate Authority (CA). AWS IoTper impostazione predefinita, i certificati sono associati al certificato CA root di Amazon.

   Linux or Unix

   ```
   sudo curl -o ./greengrass-v2-certs/AmazonRootCA1.pem https://
   www.amazontrust.com/repository/AmazonRootCA1.pem
   ```

   Windows Command Prompt (CMD)

   ```
   curl -o .\greengrass-v2-certs\AmazonRootCA1.pem https://www.amazontrust.com/
   repository/AmazonRootCA1.pem
   ```

   PowerShell

   ```
   iwr -Uri https://www.amazontrust.com/repository/AmazonRootCA1.pem -OutFile .
   \greengrass-v2-certs\AmazonRootCA1.pem
   ```

2. Se prevedi di eseguire il software AWS IoT Greengrass Core in Docker su un dispositivo diverso dal computer di sviluppo, copia i certificati sul computer host. Se SSH e SCP sono abilitati sul computer di sviluppo e sul computer host, puoi usare il `scp` comando sul tuo computer di sviluppo per trasferire i certificati. *`device-ip-address`*Sostituiscilo con l'indirizzo IP del tuo computer host.

   ```
   scp -r greengrass-v2-certs/ device-ip-address:~
   ```

Crea un file di configurazione

1. Sul computer host, crea una cartella in cui inserire il file di configurazione.

```
mkdir ./greengrass-v2-config
```

2. Utilizzate un editor di testo per creare un file di configurazione denominato `config.yaml` nella `./greengrass-v2-config` cartella.

   Ad esempio, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il`config.yaml`.

   ```
   nano ./greengrass-v2-config/config.yaml
   ```

3. Copiate il seguente contenuto YAML nel file. Questo file di configurazione parziale specifica i parametri di sistema e i parametri del nucleo di Greengrass.

   ```
   ---
   system:
     certificateFilePath: "/tmp/certs/device.pem.crt"
     privateKeyPath: "/tmp/certs/private.pem.key"
     rootCaPath: "/tmp/certs/AmazonRootCA1.pem"
     rootpath: "/greengrass/v2"
     thingName: "MyGreengrassCore"
   services:
     aws.greengrass.Nucleus:
       componentType: "NUCLEUS"
       version: "nucleus-version"
       configuration:
         awsRegion: "region"
         iotRoleAlias: "GreengrassCoreTokenExchangeRoleAlias"
         iotDataEndpoint: "device-data-prefix-ats.iot.region.amazonaws.com"
         iotCredEndpoint: "device-credentials-prefix.credentials.region.amazonaws.com"
   ```

   Quindi, sostituite i seguenti valori:

   - */tmp/certs*. La directory nel contenitore Docker in cui vengono montati i certificati scaricati all'avvio del contenitore.
   - */greengrass/v2*. La cartella principale di Greengrass che si desidera utilizzare per l'installazione. Utilizzate la variabile di `GGC_ROOT` ambiente per impostare questo valore.
   - *MyGreengrassCore*. Nome del nuovo oggetto AWS IoT.
   - *versione nucleus*. La versione del software AWS IoT Greengrass Core da installare. Questo valore deve corrispondere alla versione dell'immagine Docker o del Dockerfile che hai

scaricato. Se hai scaricato l'immagine Greengrass Docker con il `latest` tag, usala **`docker inspect`** ***`image-id`*** per vedere la versione dell'immagine.

- *`regione.`* Il Regione AWS luogo in cui hai creato AWS IoT le tue risorse. È inoltre necessario specificare lo stesso valore per la variabile di `AWS_REGION` ambiente nel file di ambiente.
- *`GreengrassCoreTokenExchangeRoleAlias`*. L'alias del ruolo di scambio di token.
- *`device-data-prefix`*. Il prefisso per il tuo endpoint di AWS IoT dati.
- *`device-credentials-prefix`*. Il prefisso per l'endpoint AWS IoT delle credenziali.

Crea un file di ambiente

Questo tutorial utilizza un file di ambiente per impostare le variabili di ambiente che verranno passate al programma di installazione del software AWS IoT Greengrass Core all'interno del contenitore Docker. Puoi anche usare l'`--env`argomento `-e` or nel `docker run` comando per impostare le variabili di ambiente nel contenitore Docker oppure puoi impostare le variabili in un `environment` blocco del file. `docker-compose.yml`

1. Usa un editor di testo per creare un file di ambiente denominato `.env`.

   Ad esempio, su un sistema basato su Linux, puoi eseguire il seguente comando per usare GNU nano per crearlo `.env` nella directory corrente.

   ```
   nano .env
   ```

2. Copiate il seguente contenuto nel file.

   ```
   GGC_ROOT_PATH=/greengrass/v2
   AWS_REGION=region
   PROVISION=false
   COMPONENT_DEFAULT_USER=ggc_user:ggc_group
   INIT_CONFIG=/tmp/config/config.yaml
   ```

   Quindi, sostituisci i seguenti valori.

   - */greengrass/v2*. Il percorso della cartella principale da utilizzare per installare il software AWS IoT Greengrass Core.
   - *`regione`*. Il Regione AWS luogo in cui hai creato AWS IoT le tue risorse. È necessario specificare lo stesso valore per il parametro `awsRegion` di configurazione nel file di configurazione.

- */tmp/config/*. La cartella in cui monti il file di configurazione all'avvio del contenitore Docker.

> **Note**
>
> È possibile impostare la variabile di `DEPLOY_DEV_TOOLS` ambiente su `true` per distribuire il componente Greengrass CLI, che consente di sviluppare componenti personalizzati all'interno del contenitore Docker. Si consiglia di utilizzare questo componente solo in ambienti di sviluppo, non in ambienti di produzione. Questo componente fornisce l'accesso a informazioni e operazioni che in genere non sono necessarie in un ambiente di produzione. Segui il principio del privilegio minimo distribuendo questo componente solo sui dispositivi principali dove ne hai bisogno.

Esegui il software AWS IoT Greengrass Core in un contenitore

Questo tutorial mostra come avviare l'immagine Docker che hai creato in un contenitore Docker. Puoi utilizzare la CLI Docker o la CLI Docker Compose per AWS IoT Greengrass eseguire l'immagine del software Core in un contenitore Docker.

Docker

- Questo tutorial mostra come avviare l'immagine Docker che hai creato in un contenitore Docker.

  ```
  docker run --rm --init -it --name docker-image \
   -v path/to/greengrass-v2-config:/tmp/config/:ro \
   -v path/to/greengrass-v2-certs:/tmp/certs:ro \
   --env-file .env \
   -p 8883 \
   your-container-image:version
  ```

  Questo comando di esempio utilizza i seguenti argomenti per docker run:

  - `--rm`. Pulisce il contenitore quando esce.
  - `--init`. Utilizza un processo di inizializzazione nel contenitore.

> **Note**
>
> L'`--init`argomento è necessario per chiudere il software AWS IoT Greengrass Core quando si arresta il contenitore Docker.

- `-it`. (Facoltativo) Esegue il contenitore Docker in primo piano come processo interattivo. Puoi sostituirlo con l'`-d`argomento per eseguire invece il contenitore Docker in modalità distaccata. Per ulteriori informazioni, consulta Detached vs foreground nella documentazione Docker.
- `--name`. Esegue un contenitore denominato `aws-iot-greengrass`
- `-v`. Monta un volume nel contenitore Docker per rendere il file di configurazione e i file di certificato disponibili per l'AWS IoT Greengrassesecuzione all'interno del contenitore.
- `--env-file`. (Facoltativo) Speciifica il file di ambiente per impostare le variabili di ambiente che verranno passate al programma di installazione del software AWS IoT Greengrass Core all'interno del contenitore Docker. Questo argomento è richiesto solo se è stato creato un file di ambiente per impostare le variabili di ambiente. Se non hai creato un file di ambiente, puoi utilizzare `--env` gli argomenti per impostare le variabili di ambiente direttamente nel comando Docker run.
- `-p`. (Facoltativo) Pubblica la porta container 8883 sulla macchina host. Questo argomento è necessario se si desidera connettersi e comunicare tramite MQTT perché AWS IoT Greengrass utilizza la porta 8883 per il traffico MQTT. Per aprire altre porte, utilizzate argomenti aggiuntivi. `-p`

> **Note**
>
> Per eseguire il contenitore Docker con maggiore sicurezza, puoi utilizzare gli `--cap-add` argomenti `--cap-drop` e per abilitare selettivamente le funzionalità Linux per il tuo contenitore. Per ulteriori informazioni, consulta Privilegi di runtime e funzionalità Linux nella documentazione Docker.

Docker Compose

1. Usa un editor di testo per creare un file Docker Compose denominato. `docker-compose.yml`

Ad esempio, su un sistema basato su Linux, puoi eseguire il seguente comando per usare GNU nano per crearlo nella directory corrente. `docker-compose.yml`

```
nano docker-compose.yml
```

> **Note**
>
> È inoltre possibile scaricare e utilizzare la versione più recente del file Compose fornito AWS da. GitHub

2. Aggiungi il seguente contenuto al file Compose. Il file si presenta in maniera simile al seguente frammento: Sostituisci:version *`your-container-namecon`* il nome della tua immagine Docker.

```
version: '3.7'

services:
  greengrass:
    init: true
    build:
      context: .
    container_name: aws-iot-greengrass
    image: your-container-name:version
    volumes:
      - /path/to/greengrass-v2-config:/tmp/config/:ro
      - /path/to/greengrass-v2-certs:/tmp/certs:ro
    env_file: .env
    ports:
      - "8883:8883"
```

I seguenti parametri in questo file Compose di esempio sono facoltativi:

- `ports`—Pubblica le porte del contenitore 8883 sul computer host. Questo parametro è necessario se si desidera connettersi e comunicare tramite MQTT perché AWS IoT Greengrass utilizza la porta 8883 per il traffico MQTT.
- `env_file`: specifica il file di ambiente per impostare le variabili di ambiente che verranno passate al programma di installazione del software AWS IoT Greengrass Core all'interno del contenitore Docker. Questo parametro è richiesto solo se è stato creato un file di

ambiente per impostare le variabili di ambiente. Se non avete creato un file di ambiente, potete utilizzare il parametro environment per impostare le variabili direttamente nel file Compose.

> **Note**
>
> Per eseguire il contenitore Docker con maggiore sicurezza, puoi utilizzare `cap_drop` e `cap_add` nel tuo file Compose per abilitare selettivamente le funzionalità Linux per il tuo contenitore. Per ulteriori informazioni, consulta Privilegi di runtime e funzionalità Linux nella documentazione Docker.

3. Esegui il comando seguente per avviare il contenitore.

```
docker-compose -f docker-compose.yml up
```

Passaggi successivi

AWS IoT GreengrassIl software principale è ora in esecuzione in un contenitore Docker. Esegui il comando seguente per recuperare l'ID del contenitore attualmente in esecuzione.

```
docker ps
```

È quindi possibile eseguire il comando seguente per accedere al contenitore ed esplorare il software AWS IoT Greengrass Core in esecuzione all'interno del contenitore.

```
docker exec -it container-id /bin/bash
```

Per informazioni sulla creazione di un componente semplice, consulta Fase 4: Sviluppa e testa un componente sul tuo dispositivo in Tutorial: Nozioni di base su AWS IoT Greengrass V2

> **Note**
>
> Quando esegui comandi all'interno del contenitore Docker, tali comandi non vengono registrati nei log Docker. `docker exec` Per registrare i comandi nei log Docker, collega una shell interattiva al contenitore Docker. Per ulteriori informazioni, consulta Collega una shell interattiva al contenitore Docker.

Il file di registro AWS IoT Greengrass Core viene chiamato `greengrass.log` e si trova in. `/greengrass/v2/logs` Nella stessa directory si trovano anche i file di registro dei componenti. Per copiare i log di Greengrass in una directory temporanea sull'host, esegui il seguente comando:

```
docker cp container-id:/greengrass/v2/logs /tmp/logs
```

Se desideri mantenere i log dopo l'uscita o la rimozione di un contenitore, ti consigliamo di collegare solo la `/greengrass/v2/logs` directory alla directory dei log temporanei sull'host invece di montare l'intera directory Greengrass. Per ulteriori informazioni, consulta Mantieni i log Greengrass all'esterno del contenitore Docker.

Per fermare un contenitore Docker in esecuzione, esegui o. AWS IoT Greengrass `docker stop docker-compose -f docker-compose.yml stop` Questa azione invia `SIGTERM` al processo Greengrass e chiude tutti i processi associati che sono stati avviati nel contenitore. Il contenitore Docker viene inizializzato con l'`docker-init`eseguibile come processo PID 1, il che aiuta a rimuovere eventuali processi zombie rimanenti. Per ulteriori informazioni, consulta Specificare un processo di inizializzazione nella documentazione di Docker.

Per informazioni sulla risoluzione dei problemi relativi all'esecuzione AWS IoT Greengrass in un contenitore Docker, consulta. Risoluzione dei problemi di AWS IoT Greengrass in un container Docker

Risoluzione dei problemi di AWS IoT Greengrass in un container Docker

Utilizza le seguenti informazioni per aiutarti a risolvere i problemi relativi all'esecuzione AWS IoT Greengrass in un contenitore Docker e per eseguire il debug dei problemi AWS IoT Greengrass nel contenitore Docker.

Argomenti

- Risoluzione dei problemi relativi all'esecuzione del contenitore Docker
- Debug di AWS IoT Greengrass in un container Docker

Risoluzione dei problemi relativi all'esecuzione del contenitore Docker

Utilizza le informazioni riportate di seguito per risolvere i problemi più comuni con l'esecuzione di AWS IoT Greengrass in un container Docker.

Argomenti

Errore: impossibile eseguire un accesso interattivo da un dispositivo non TTY

Questo errore può verificarsi quando si esegue il `aws ecr get-login-password` comando. Assicurati di aver installato la versione 2 o la AWS CLI versione 1 più recente. Ti consigliamo di utilizzare la AWS CLI versione 2. Per ulteriori informazioni, consulta Installazione dell'AWS CLI nella Guida per l'utente di AWS Command Line Interface.

Errore: opzioni sconosciute: - no-include-email

Questo errore può verificarsi quando si esegue il `aws ecr get-login` comando. Verifica che sia installata la versione AWS CLI più recente (per esempio, esegui: `pip install awscli --upgrade --user`). Per ulteriori informazioni, vedere Installazione di AWS Command Line Interface su Microsoft Windows nella Guida per l'AWS Command Line Interfaceutente.

Errore: un firewall sta bloccando la condivisione di file tra finestre e contenitori.

Potresti ricevere questo errore o un `Firewall Detected` messaggio quando esegui Docker su un computer Windows. Questo errore può inoltre verificarsi se sei connesso a una rete privata virtuale (VPN, Virtual Private Network) e le impostazioni di rete impediscono il montaggio dell'unità condivisa. In tal caso, disattivare la rete VPN e riavviare il container Docker.

Errore: si è verificato un errore (AccessDeniedException) durante la chiamata dell' GetAuthorizationToken operazione: User: arn:aws:iam:: ***`account-id:user/`*** is <user-name>not authorized to perform: ecr: on resource: * GetAuthorizationToken

Potresti ricevere questo errore durante l'esecuzione del `aws ecr get-login-password` comando se non disponi di autorizzazioni sufficienti per accedere a un repository Amazon ECR. Per ulteriori informazioni, consulta Esempi di policy per i repository di Amazon ECR e Accesso a un repository Amazon ECR nella Amazon ECR User Guide.

Errore: hai raggiunto il limite di pull rate

Docker Hub limita il numero di richieste pull che gli utenti anonimi e gratuiti di Docker Hub possono effettuare. Se superi i limiti di velocità per le pull request anonime o gratuite degli utenti, ricevi uno dei seguenti errori:

```
ERROR: toomanyrequests: Too Many Requests.
```

```
You have reached your pull rate limit.
```

Per risolvere questi errori, puoi attendere qualche ora prima di provare un'altra pull request. Se prevedi di inviare costantemente un numero elevato di pull request, consulta il sito Web di Docker Hub per informazioni sui limiti di velocità e sulle opzioni per l'autenticazione e l'aggiornamento del tuo account Docker.

Debug di AWS IoT Greengrass in un container Docker

Per il debug dei problemi relativi a un container Docker, puoi rendere persistenti i log di runtime Greengrass o collegare una shell interattiva al container Docker.

Mantieni i log Greengrass all'esterno del contenitore Docker

Dopo aver fermato un AWS IoT Greengrass contenitore, puoi usare il seguente `docker cp` comando per copiare i log di Greengrass dal contenitore Docker in una directory di log temporanea.

```
docker cp container-id:/greengrass/v2/logs /tmp/logs
```

Per mantenere i log anche dopo l'uscita o la rimozione di un contenitore, è necessario eseguire il contenitore Docker dopo aver montato la AWS IoT Greengrass directory tramite bind. */greengrass/v2*`/logs`

Per montare in modo bind-mount la */greengrass/v2*`/logs` directory, esegui una delle seguenti operazioni quando esegui un nuovo contenitore Docker. AWS IoT Greengrass

- Includi `-v` */tmp/logs*`:`*/greengrass/v2*`/logs:ro` nel tuo comando. `docker run`

  Modifica il `volumes` blocco nel file Compose per includere la riga seguente prima di eseguire il `docker-compose up` comando.

```
volumes:
 - /tmp/logs:/greengrass/v2/logs:ro
```

Puoi quindi controllare i log */tmp/logs* sul tuo host per vedere i log di Greengrass AWS IoT Greengrass mentre è in esecuzione all'interno del contenitore Docker.

Per informazioni sull'esecuzione dei contenitori Greengrass Docker, vedere e Esegui AWS IoT Greengrass in Docker con provisioning manuale Esegui AWS IoT Greengrass in Docker con provisioning automatico

Collega una shell interattiva al contenitore Docker

Quando esegui comandi all'interno del contenitore Docker, tali comandi non vengono acquisiti nei log Docker. `docker exec` La registrazione dei comandi nei registri Docker può aiutarti a esaminare lo stato del contenitore Greengrass Docker. Esegui una di queste operazioni:

- Esegui il comando seguente in un terminale separato per collegare l'input, l'output e l'errore standard del tuo terminale al contenitore in esecuzione. Ciò consente di visualizzare e controllare il contenitore Docker dal terminale corrente.

  ```
  docker attach container-id
  ```

- Esegui il comando seguente in un terminale separato. Ciò consente di eseguire i comandi in modalità interattiva, anche se il contenitore non è collegato.

  ```
  docker exec -it container-id sh -c "command > /proc/1/fd/1"
  ```

Per una AWS IoT Greengrass risoluzione generale dei problemi, vedereRisoluzione dei problemi.

Configurare il software AWS IoT Greengrass Core

Il software AWS IoT Greengrass Core offre opzioni che è possibile utilizzare per configurare il software. È possibile creare distribuzioni per configurare il software AWS IoT Greengrass Core su ciascun dispositivo principale.

Argomenti

- Implementa il componente Greengrass nucleus
- Configurare il nucleo Greengrass come servizio di sistema
- Controlla l'allocazione della memoria con le opzioni JVM
- Configurare l'utente che esegue i componenti
- Configura i limiti delle risorse di sistema per i componenti
- Connessione alla porta 443 o tramite un proxy di rete
- Utilizza un certificato del dispositivo firmato da una CA privata
- Configurare i timeout MQTT e le impostazioni della cache

Implementa il componente Greengrass nucleus

AWS IoT Greengrass fornisce il software AWS IoT Greengrass Core come componente che puoi distribuire sui tuoi dispositivi principali Greengrass. Puoi creare una distribuzione per applicare la stessa configurazione a più dispositivi core Greengrass. Per ulteriori informazioni, consulta Nucleo Greengrass e Aggiornamento del software AWS IoT Greengrass Core (OTA).

Configurare il nucleo Greengrass come servizio di sistema

È necessario configurare il software AWS IoT Greengrass Core come servizio di sistema nel sistema init del dispositivo per effettuare le seguenti operazioni:

- Avvia il software AWS IoT Greengrass Core all'avvio del dispositivo. Questa è una buona pratica se gestisci grandi flotte di dispositivi.
- Installa ed esegui i componenti del plug-in. Diversi componenti AWS forniti sono componenti plug-in, che consentono loro di interfacciarsi direttamente con il nucleo Greengrass. Per ulteriori informazioni sui tipi di componenti, vedere. Tipi di componenti
- Applica gli aggiornamenti over-the-air (OTA) al software AWS IoT Greengrass Core del dispositivo principale. Per ulteriori informazioni, consulta Aggiornamento del software AWS IoT Greengrass Core (OTA).
- Abilita i componenti per riavviare il software AWS IoT Greengrass Core o il dispositivo principale quando una distribuzione aggiorna il componente a una nuova versione o aggiorna determinati parametri di configurazione. Per ulteriori informazioni, consulta la fase del ciclo di vita di bootstrap.

> **Important**
>
> Nei dispositivi Windows Core, è necessario configurare il software AWS IoT Greengrass Core come servizio di sistema.

Argomenti

Configurate il nucleo come servizio di sistema (Linux)

I dispositivi Linux supportano diversi sistemi di init, come initd, systemd e SystemV. L'`--setup-system-service true`argomento viene utilizzato quando si installa il software AWS IoT Greengrass Core per avviare il nucleus come servizio di sistema e configurarlo per l'avvio all'avvio del dispositivo. Il programma di installazione configura il software AWS IoT Greengrass Core come servizio di sistema con systemd.

È anche possibile configurare manualmente il nucleo per l'esecuzione come servizio di sistema. L'esempio seguente è un file di servizio per systemd.

```
[Unit]
Description=Greengrass Core

[Service]
Type=simple
PIDFile=/greengrass/v2/alts/loader.pid
RemainAfterExit=no
Restart=on-failure
RestartSec=10
ExecStart=/bin/sh /greengrass/v2/alts/current/distro/bin/loader

[Install]
WantedBy=multi-user.target
```

Dopo aver configurato il servizio di sistema, è possibile eseguire i seguenti comandi per configurare l'avvio del dispositivo all'avvio e per avviare o arrestare il software AWS IoT Greengrass Core.

- Per verificare lo stato del servizio (systemd)

```
sudo systemctl status greengrass.service
```

- Per consentire al nucleo di avviarsi all'avvio del dispositivo.

```
sudo systemctl enable greengrass.service
```

- Per impedire che il nucleo si avvii all'avvio del dispositivo.

```
sudo systemctl disable greengrass.service
```

- Per avviare il software AWS IoT Greengrass Core.

```
sudo systemctl start greengrass.service
```

- Per interrompere il software AWS IoT Greengrass Core.

```
sudo systemctl stop greengrass.service
```

Configurare il nucleo come servizio di sistema (Windows)

L'`--setup-system-service true`argomento viene utilizzato quando si installa il software AWS IoT Greengrass Core per avviare il nucleo come servizio Windows e configurarlo per l'avvio all'avvio del dispositivo.

Dopo aver configurato il servizio, puoi eseguire i seguenti comandi per configurare l'avvio del dispositivo all'avvio e per avviare o arrestare il software AWS IoT Greengrass Core. È necessario eseguire il prompt dei comandi o PowerShell come amministratore per eseguire questi comandi.

Windows Command Prompt (CMD)

- Per verificare lo stato del servizio

```
sc query "greengrass"
```

- Per consentire al nucleo di avviarsi all'avvio del dispositivo.

```
sc config "greengrass" start=auto
```

- Per impedire che il nucleo si avvii all'avvio del dispositivo.

```
sc config "greengrass" start=disabled
```

- Per avviare il software AWS IoT Greengrass Core.

```
sc start "greengrass"
```

- Per interrompere il software AWS IoT Greengrass Core.

```
sc stop "greengrass"
```

> **Note**
>
> Sui dispositivi Windows, il software AWS IoT Greengrass Core ignora questo segnale di spegnimento mentre interrompe i processi dei componenti Greengrass. Se il software AWS IoT Greengrass Core ignora il segnale di spegnimento quando esegui questo comando, attendi qualche secondo e riprova.

PowerShell

- Per verificare lo stato del servizio

```
Get-Service -Name "greengrass"
```

- Per consentire al nucleo di avviarsi all'avvio del dispositivo.

```
Set-Service -Name "greengrass" -Status stopped -StartupType automatic
```

- Per impedire che il nucleo si avvii all'avvio del dispositivo.

```
Set-Service -Name "greengrass" -Status stopped -StartupType disabled
```

- Per avviare il software AWS IoT Greengrass Core.

```
Start-Service -Name "greengrass"
```

- Per interrompere il software AWS IoT Greengrass Core.

```
Stop-Service -Name "greengrass"
```

> **Note**
>
> Sui dispositivi Windows, il software AWS IoT Greengrass Core ignora questo segnale di spegnimento mentre interrompe i processi dei componenti Greengrass. Se il software AWS IoT Greengrass Core ignora il segnale di spegnimento quando esegui questo comando, attendi qualche secondo e riprova.

Controlla l'allocazione della memoria con le opzioni JVM

Se utilizzi AWS IoT Greengrass un dispositivo con memoria limitata, puoi utilizzare le opzioni della macchina virtuale Java (JVM) per controllare la dimensione massima dell'heap, le modalità di raccolta dei rifiuti e le opzioni del compilatore, che controllano la quantità di memoria utilizzata dal software Core. AWS IoT Greengrass La dimensione dell'heap nella JVM determina la quantità di memoria che un'applicazione può utilizzare prima che si verifichi la raccolta dei dati indesiderati o prima che l'applicazione esaurisca la memoria. La dimensione heap massima specifica la quantità massima di memoria che JVM può allocare durante l'espansione dell'heap nel quadro di un'attività intensa.

Per controllare l'allocazione della memoria, create una nuova distribuzione o modificate una distribuzione esistente che include il componente nucleus e specificate le opzioni JVM nel parametro di configurazione nella configurazione del componente nucleus. `jvmOptions`

A seconda delle esigenze, è possibile eseguire il software AWS IoT Greengrass Core con un'allocazione di memoria ridotta o con un'allocazione di memoria minima.

Allocazione di memoria ridotta

Per eseguire il software AWS IoT Greengrass Core con un'allocazione di memoria ridotta, si consiglia di utilizzare il seguente esempio di configurazione merge update per impostare le opzioni JVM nella configurazione del nucleo:

```
{
  "jvmOptions": "-Xmx64m -XX:+UseSerialGC -XX:TieredStopAtLevel=1"
}
```

Allocazione minima di memoria

Per eseguire il software AWS IoT Greengrass Core con un'allocazione di memoria minima, si consiglia di utilizzare il seguente esempio di configurazione merge update per impostare le opzioni JVM nella configurazione del nucleo:

```
{
  "jvmOptions": "-Xmx32m -XX:+UseSerialGC -Xint"
}
```

Questi esempi di aggiornamenti di fusione della configurazione utilizzano le seguenti opzioni JVM:

`-Xmx`*NN*`m`

Imposta la dimensione massima dell'heap JVM.

Per un'allocazione di memoria ridotta, utilizzare `-Xmx64m` come valore iniziale per limitare la dimensione dell'heap a 64 MB. Per un'allocazione minima di memoria, utilizzare `-Xmx32m` come valore iniziale per limitare la dimensione dell'heap a 32 MB.

Puoi aumentare o diminuire il `-Xmx` valore in base alle tue effettive esigenze; tuttavia, ti consigliamo vivamente di non impostare la dimensione massima dell'heap al di sotto di 16 MB. Se la dimensione massima dell'heap è troppo bassa per il tuo ambiente, il software AWS IoT Greengrass Core potrebbe riscontrare errori imprevisti a causa della memoria insufficiente.

`-XX:+UseSerialGC`

Speciifica di utilizzare la raccolta seriale dei rifiuti per lo spazio heap JVM. Il serial garbage collector è più lento, ma utilizza meno memoria rispetto ad altre implementazioni di JVM Garbage Collector.

`-XX:TieredStopAtLevel=1`

Indica alla JVM di utilizzare il compilatore Java (JIT) una volta. just-in-time Poiché il codice compilato JIT utilizza spazio nella memoria del dispositivo, l'utilizzo del compilatore JIT più di una volta consuma più memoria di una singola compilazione.

`-Xint`

Indica alla JVM di non utilizzare il compilatore (JIT). just-in-time La JVM viene invece eseguita in modalità solo interpretata. Questa modalità è più lenta dell'esecuzione del codice compilato JIT; tuttavia, il codice compilato non utilizza spazio in memoria.

Per informazioni sulla creazione di aggiornamenti di fusione della configurazione, consulta. Aggiornamento delle configurazioni dei componenti

Configurare l'utente che esegue i componenti

Il software AWS IoT Greengrass Core può eseguire i processi dei componenti come utente e gruppo di sistema diversi da quello che esegue il software. Ciò aumenta la sicurezza, poiché è possibile eseguire il software AWS IoT Greengrass Core come utente root o come utente amministratore, senza concedere tali autorizzazioni ai componenti in esecuzione sul dispositivo principale.

La tabella seguente indica i tipi di componenti che il software AWS IoT Greengrass Core può eseguire come utente specificato. Per ulteriori informazioni, consulta Tipi di componenti.

Tipo di componente	Configura l'utente del componente
Nucleo	 No
Plug-in	 No
Generico	 Sì
Lambda (non containerizzata)	 Sì

Tipo di componente	Configura l'utente del componente
Lambda (containerizzata)	 Sì

È necessario creare l'utente del componente prima di poterlo specificare in una configurazione di distribuzione. Nei dispositivi basati su Windows, è inoltre necessario memorizzare il nome utente e la password dell'utente nell'istanza di gestione delle credenziali dell'account. LocalSystem Per ulteriori informazioni, consulta Configura un utente del componente sui dispositivi Windows.

Quando si configura l'utente del componente su un dispositivo basato su Linux, è possibile facoltativamente specificare anche un gruppo. Specificate l'utente e il gruppo separati da due punti (`:`) nel seguente formato:. *`user`*`:`*`group`* Se non specifichi un gruppo, il software AWS IoT Greengrass Core utilizza per impostazione predefinita il gruppo principale dell'utente. È possibile utilizzare il nome o l'ID per identificare l'utente e il gruppo.

Sui dispositivi basati su Linux, puoi anche eseguire componenti come utente di sistema inesistente, chiamato anche utente sconosciuto, per aumentare la sicurezza. Un processo Linux può segnalare qualsiasi altro processo eseguito dallo stesso utente. Un utente sconosciuto non esegue altri processi, quindi puoi eseguire i componenti come utente sconosciuto per evitare che i componenti segnalino altri componenti sul dispositivo principale. Per eseguire i componenti come utente sconosciuto, specifica un ID utente che non esiste sul dispositivo principale. Puoi anche specificare un ID di gruppo che non esiste da eseguire come gruppo sconosciuto.

È possibile configurare l'utente per ogni componente e per ogni dispositivo principale.

- Configurazione per un componente

  È possibile configurare ogni componente in modo che venga eseguito con un utente specifico per quel componente. Quando si crea una distribuzione, è possibile specificare l'utente per ogni componente nella `runWith` configurazione di quel componente. Il software AWS IoT Greengrass Core esegue i componenti come utente specificato se li configurate. Altrimenti, per impostazione predefinita, esegue i componenti come utente predefinito configurato per il dispositivo principale. Per ulteriori informazioni sulla specificazione dell'utente del componente nella configurazione di distribuzione, vedere il parametro di `runWith`configurazione in. Creare distribuzione

- Configura l'utente predefinito per un dispositivo principale

  È possibile configurare un utente predefinito utilizzato dal software AWS IoT Greengrass Core per eseguire i componenti. Quando il software AWS IoT Greengrass Core esegue un componente, verifica se è stato specificato un utente per quel componente e lo utilizza per eseguire il componente. Se il componente non specifica un utente, il software AWS IoT Greengrass Core esegue il componente come utente predefinito configurato per il dispositivo principale. Per ulteriori informazioni, consulta Configura l'utente predefinito del componente.

> **Note**
>
> Sui dispositivi basati su Windows, è necessario specificare almeno un utente predefinito per eseguire i componenti.
>
> Sui dispositivi basati su Linux, si applicano le seguenti considerazioni se non si configura un utente per l'esecuzione dei componenti:
>
> - Se esegui il software AWS IoT Greengrass Core come root, il software non eseguirà componenti. È necessario specificare un utente predefinito per eseguire i componenti se si esegue come root.
> - Se esegui il software AWS IoT Greengrass Core come utente non root, il software esegue i componenti come tale utente.

Argomenti

- Configura un utente del componente sui dispositivi Windows
- Configura l'utente predefinito del componente

Configura un utente del componente sui dispositivi Windows

Per configurare un utente del componente su un dispositivo basato su Windows

1. Crea l'utente del componente nell' LocalSystem account sul dispositivo.

   ```
   net user /add component-user password
   ```

2. Utilizza l' PsExec utilità Microsoft per memorizzare il nome utente e la password per l'utente del componente nell'istanza di Credential Manager per l' LocalSystem account.

```
psexec -s cmd /c cmdkey /generic:component-user /user:component-user /pass:password
```

> **Note**
>
> Sui dispositivi basati su Windows, l' LocalSystem account esegue il Greengrass nucleus ed è necessario utilizzare l' PsExec utilità per memorizzare le informazioni utente del componente nell'account. LocalSystem L'utilizzo dell'applicazione Credential Manager archivia queste informazioni nell'account Windows dell'utente attualmente connesso, anziché nell'account. LocalSystem

Configura l'utente predefinito del componente

È possibile utilizzare una distribuzione per configurare l'utente predefinito su un dispositivo principale. In questa distribuzione, aggiorni la configurazione del componente nucleus.

> **Note**
>
> Puoi anche impostare l'utente predefinito quando installi il software AWS IoT Greengrass Core con l'`--component-default-user`opzione. Per ulteriori informazioni, consulta Installare il software AWS IoT Greengrass Core..

Crea una distribuzione che specifichi il seguente aggiornamento di configurazione per il `aws.greengrass.Nucleus` componente.

Linux

```
{
  "runWithDefault": {
    "posixUser": "ggc_user:ggc_group"
  }
}
```

Windows

```
{
  "runWithDefault": {
    "windowsUser": "ggc_user"
  }
}
```

> **Note**
>
> L'utente specificato deve esistere e il nome utente e la password per questo utente devono essere archiviati nell'istanza di gestione delle credenziali dell' LocalSystem account sul dispositivo Windows. Per ulteriori informazioni, consulta Configura un utente del componente sui dispositivi Windows.

L'esempio seguente definisce una distribuzione per un dispositivo basato su Linux che si configura `ggc_user` come utente e `ggc_group` gruppo predefiniti. L'aggiornamento della `merge` configurazione richiede un oggetto JSON serializzato.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.3",
      "configurationUpdate": {
        "merge": "{\"runWithDefault\":{\"posixUser\":\"ggc_user:ggc_group\"}}"
      }
    }
  }
}
```

Configura i limiti delle risorse di sistema per i componenti

> **Note**
>
> Questa funzionalità è disponibile per la versione 2.4.0 e successive del componente Greengrass nucleus. AWS IoT Greengrass attualmente non supporta questa funzionalità sui dispositivi Windows core.

È possibile configurare la quantità massima di utilizzo di CPU e RAM che i processi di ciascun componente possono utilizzare sul dispositivo principale.

La tabella seguente mostra i tipi di componenti che supportano i limiti delle risorse di sistema. Per ulteriori informazioni, consulta Tipi di componenti.

Tipo di componente	Configura i limiti delle risorse di sistema
Nucleo	 No
Plug-in	 No
Generico	 Sì
Lambda (non containerizzata)	 Sì
Lambda (containerizzata)	 No

> **Important**
>
> I limiti delle risorse di sistema non sono supportati quando si esegue il software AWS IoT Greengrass Core in un contenitore Docker.

Puoi configurare i limiti delle risorse di sistema per ogni componente e per ogni dispositivo principale.

- Configurazione per un componente

  È possibile configurare ogni componente con limiti di risorse di sistema specifici per quel componente. Quando si crea una distribuzione, è possibile specificare i limiti delle risorse di sistema per ogni componente della distribuzione. Se il componente supporta i limiti delle risorse di sistema, il software AWS IoT Greengrass Core applica i limiti ai processi del componente. Se non specificate i limiti delle risorse di sistema per un componente, il software AWS IoT Greengrass Core utilizza tutte le impostazioni predefinite che avete configurato per il dispositivo principale. Per ulteriori informazioni, consulta Creare distribuzione.
- Configura le impostazioni predefinite per un dispositivo principale

  È possibile configurare i limiti di risorse di sistema predefiniti che il software AWS IoT Greengrass Core applica ai componenti che supportano tali limiti. Quando il software AWS IoT Greengrass Core esegue un componente, applica i limiti delle risorse di sistema specificati per quel componente. Se quel componente non specifica i limiti delle risorse di sistema, il software AWS IoT Greengrass Core applica i limiti di risorse di sistema predefiniti configurati per il dispositivo principale. Se non specificate i limiti predefiniti delle risorse di sistema, per impostazione predefinita il software AWS IoT Greengrass Core non applica alcun limite alle risorse di sistema. Per ulteriori informazioni, consulta Configura i limiti predefiniti delle risorse di sistema.

Configura i limiti predefiniti delle risorse di sistema

È possibile implementare il componente Greengrass nucleus per configurare i limiti di risorse di sistema predefiniti per un dispositivo principale. Per configurare i limiti predefiniti delle risorse di sistema, create una distribuzione che specifichi il seguente aggiornamento della configurazione per il componente. `aws.greengrass.Nucleus`

```
{
  "runWithDefault": {
    "systemResourceLimits": {
```

```
      "cpu": cpuTimeLimit,
      "memory": memoryLimitInKb
    }
  }
}
```

L'esempio seguente definisce una distribuzione che configura il limite di tempo della CPU su2, che equivale al 50% di utilizzo su un dispositivo con 4 core CPU. Questo esempio configura anche l'utilizzo della memoria su 100 MB.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.3",
      "configurationUpdate": {
        "merge": "{\"runWithDefault\":{\"systemResourceLimits\":\"cpus\":2,\"memory
\":102400}}}"
      }
    }
  }
}
```

Connessione alla porta 443 o tramite un proxy di rete

AWS IoT Greengrass i dispositivi principali comunicano AWS IoT Core utilizzando il protocollo di messaggistica MQTT con autenticazione client TLS. Per convenzione, il protocollo MQTT su TLS utilizza la porta 8883. Tuttavia, come misura di sicurezza, gli ambienti restrittivi potrebbero limitare il traffico in entrata e in uscita a un numero ridotto di porte TCP. Ad esempio, un firewall aziendale potrebbe aprire la porta 443 per il traffico HTTPS, ma chiudere le altre porte utilizzate per i protocolli meno comuni, come la porta 8883 per il traffico MQTT. Altri ambienti restrittivi potrebbero richiedere che tutto il traffico passi attraverso un proxy prima di connettersi a Internet.

> **Note**
>
> I dispositivi core Greengrass che eseguono il componente Greengrass nucleus v2.0.3 e versioni precedenti utilizzano la porta 8443 per connettersi all'endpoint del piano dati. AWS IoT Greengrass Questi dispositivi devono essere in grado di connettersi a questo endpoint sulla porta 8443. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Per abilitare la comunicazione in questi scenari, AWS IoT Greengrass fornisce le seguenti opzioni di configurazione:

- Comunicazione MQTT tramite la porta 443. Se la rete consente le connessioni alla porta 443, è possibile configurare il dispositivo principale Greengrass in modo che utilizzi la porta 443 per il traffico MQTT anziché la porta predefinita 8883. Si può trattare di una connessione diretta alla porta 443 o di una connessione tramite un server proxy di rete. A differenza della configurazione predefinita, che utilizza l'autenticazione client basata su certificati, MQTT sulla porta 443 utilizza il ruolo del servizio del dispositivo per l'autenticazione.

  Per ulteriori informazioni, consulta Configura MQTT sulla porta 443.
- Comunicazione HTTPS tramite la porta 443. Per impostazione predefinita, il software AWS IoT Greengrass Core invia il traffico HTTPS sulla porta 8443, ma è possibile configurarlo per utilizzare la porta 443. AWS IoT Greengrass utilizza l'estensione TLS Application Layer Protocol Network (ALPN) per abilitare questa connessione. Come nella configurazione predefinita, HTTPS sulla porta 443 utilizza l'autenticazione client basata su certificati.

  > **Important**
  >
  > Per utilizzare ALPN e abilitare la comunicazione HTTPS sulla porta 443, il dispositivo principale deve eseguire Java 8 update 252 o versione successiva. Tutti gli aggiornamenti di Java versione 9 e successive supportano anche ALPN.

  Per ulteriori informazioni, consulta Configura HTTPS sulla porta 443.
- Connessione tramite un proxy di rete. È possibile configurare un server proxy di rete che funga da intermediario per la connessione al dispositivo principale Greengrass. AWS IoT Greengrass supporta l'autenticazione di base per i proxy HTTP e HTTPS.

  I dispositivi core Greengrass devono eseguire Greengrass nucleus v2.5.0 o versione successiva per utilizzare i proxy HTTPS.

  Il software AWS IoT Greengrass Core trasmette la configurazione del proxy ai componenti tramite le variabili,, e di ambiente. `ALL_PROXY` `HTTP_PROXY` `HTTPS_PROXY` `NO_PROXY` I componenti devono utilizzare queste impostazioni per connettersi tramite il proxy. I componenti utilizzano librerie comuni (come boto3, cURL e il `requests` pacchetto python) che in genere utilizzano queste variabili di ambiente per impostazione predefinita per effettuare connessioni. Se un componente specifica anche queste variabili di ambiente, AWS IoT Greengrass non le sovrascrive.

Per ulteriori informazioni, consulta Configurare un proxy di rete.

Configura MQTT sulla porta 443

È possibile configurare MQTT tramite la porta 443 su dispositivi core esistenti o quando si installa il software AWS IoT Greengrass Core su un nuovo dispositivo core.

Argomenti

- Configurate MQTT tramite la porta 443 su dispositivi core esistenti
- Configurare MQTT sulla porta 443 durante l'installazione

Configurate MQTT tramite la porta 443 su dispositivi core esistenti

È possibile utilizzare una distribuzione per configurare MQTT sulla porta 443 su un dispositivo single core o su un gruppo di dispositivi core. In questa distribuzione, aggiorni la configurazione del componente nucleus. Il nucleo si riavvia quando si aggiorna la configurazione. `mqtt`

Per configurare MQTT sulla porta 443, create una distribuzione che specifichi il seguente aggiornamento della configurazione per il componente. `aws.greengrass.Nucleus`

```
{
  "mqtt": {
    "port": 443
  }
}
```

L'esempio seguente definisce una distribuzione che configura MQTT sulla porta 443. L'aggiornamento della `merge` configurazione richiede un oggetto JSON serializzato.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.3",
      "configurationUpdate": {
        "merge": "{\"mqtt\":{\"port\":443}}"
      }
    }
  }
}
```

Configurare MQTT sulla porta 443 durante l'installazione

È possibile configurare MQTT sulla porta 443 quando si installa il software AWS IoT Greengrass Core su un dispositivo principale. Utilizzate l'argomento `--init-config` installer per configurare MQTT sulla porta 443. È possibile specificare questo argomento quando si esegue l'installazione con il provisioning manuale, il provisioning del parco veicoli o il provisioning personalizzato.

Configura HTTPS sulla porta 443

Questa funzionalità richiede la Nucleo Greengrass versione 2.0.4 o successiva.

È possibile configurare HTTPS tramite la porta 443 su dispositivi core esistenti o quando si installa il software AWS IoT Greengrass Core su un nuovo dispositivo core.

Argomenti

- Configura HTTPS tramite la porta 443 sui dispositivi principali esistenti
- Configura HTTPS sulla porta 443 durante l'installazione

Configura HTTPS tramite la porta 443 sui dispositivi principali esistenti

Puoi utilizzare una distribuzione per configurare HTTPS sulla porta 443 su un dispositivo single core o su un gruppo di dispositivi core. In questa distribuzione, aggiorni la configurazione del componente nucleus.

Per configurare HTTPS sulla porta 443, crea una distribuzione che specifichi il seguente aggiornamento della configurazione per il componente. `aws.greengrass.Nucleus`

```
{
  "greengrassDataPlanePort": 443
}
```

L'esempio seguente definisce una distribuzione che configura HTTPS sulla porta 443. L'aggiornamento della `merge` configurazione richiede un oggetto JSON serializzato.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.3",
      "configurationUpdate": {
        "merge": "{\"greengrassDataPlanePort\":443}"
```

```
      }
    }
  }
}
```

Configura HTTPS sulla porta 443 durante l'installazione

È possibile configurare HTTPS sulla porta 443 quando si installa il software AWS IoT Greengrass Core su un dispositivo principale. Utilizzate l'argomento `--init-config` installer per configurare HTTPS sulla porta 443. È possibile specificare questo argomento quando si esegue l'installazione con il provisioning manuale, il provisioning delparco veicoli o il provisioning personalizzato.

Configurare un proxy di rete

Segui una procedura in questa sezione per configurare i dispositivi core Greengrass per la connessione a Internet tramite un proxy di rete HTTP o HTTPS. Per ulteriori informazioni sugli endpoint e le porte utilizzati dai dispositivi principali, consulta. Consenti il traffico dei dispositivi tramite un proxy o un firewall

Important

Se il dispositivo principale esegue una versione del nucleo di Greengrass precedente alla v2.4.0, il ruolo del dispositivo deve consentire le seguenti autorizzazioni per utilizzare un proxy di rete:

- `iot:Connect`
- `iot:Publish`
- `iot:Receive`
- `iot:Subscribe`

Ciò è necessario perché il dispositivo utilizza le AWS credenziali del servizio di scambio di token per autenticare le connessioni MQTT. AWS IoT Il dispositivo utilizza MQTT per ricevere e installare distribuzioni da Cloud AWS, quindi il dispositivo non funzionerà a meno che non si definiscano queste autorizzazioni in base al suo ruolo. I dispositivi utilizzano in genere certificati X.509 per autenticare le connessioni MQTT, ma i dispositivi non possono eseguire questa operazione per l'autenticazione quando utilizzano un proxy.
Per ulteriori informazioni su come configurare il ruolo del dispositivo, consulta. Autorizza i dispositivi principali a interagire conAWSservizi

Argomenti

Configurare un proxy di rete sui dispositivi principali esistenti

È possibile utilizzare una distribuzione per configurare un proxy di rete su un dispositivo a core singolo o su un gruppo di dispositivi principali. In questa distribuzione, aggiorni la configurazione del componente nucleus. Il nucleo si riavvia quando si aggiorna la configurazione. `networkProxy`

Per configurare un proxy di rete, crea una distribuzione per il `aws.greengrass.Nucleus` componente che unisca il seguente aggiornamento di configurazione. Questo aggiornamento di configurazione contiene l'oggetto NetworkProxy.

```
{
  "networkProxy": {
    "noProxyAddresses": "http://192.168.0.1,www.example.com",
    "proxy": {
      "url": "https://my-proxy-server:1100"
    }
  }
}
```

L'esempio seguente definisce una distribuzione che configura un proxy di rete. L'aggiornamento della `merge` configurazione richiede un oggetto JSON serializzato.

```
{
  "components": {
    "aws.greengrass.Nucleus": {
      "version": "2.12.3",
      "configurationUpdate": {
        "merge": "{\"networkProxy\":{\"noProxyAddresses\":
\"http://192.168.0.1,www.example.com\",\"proxy\":{\"url\":\"https://my-proxy-
server:1100\",\"username\":\"Mary_Major\",\"password\":\"pass@word1357\"}}}"
      }
    }
  }
```

```
}
```

Configurare un proxy di rete durante l'installazione

È possibile configurare un proxy di rete quando si installa il software AWS IoT Greengrass Core su un dispositivo principale. Utilizzate l'argomento `--init-config` installer per configurare il proxy di rete. È possibile specificare questo argomento quando si esegue l'installazione con il provisioning manuale, il provisioning del parco veicoli o il provisioning personalizzato.

Abilita il dispositivo principale in modo che consideri attendibile un proxy HTTPS

Quando configuri un dispositivo principale per utilizzare un proxy HTTPS, devi aggiungere la catena di certificati del server proxy a quella del dispositivo principale per consentirgli di considerare attendibile il proxy HTTPS. In caso contrario, il dispositivo principale potrebbe riscontrare errori quando tenta di indirizzare il traffico attraverso il proxy. Aggiungi il certificato CA del server proxy al file di certificato CA principale Amazon del dispositivo principale.

Per consentire al dispositivo principale di considerare attendibile il proxy HTTPS

1. Trova il file del certificato Amazon root CA sul dispositivo principale.

   - Se hai installato il software AWS IoT Greengrass Core con provisioning automatico, il file di certificato CA root di Amazon esiste in*/greengrass/v2*`/rootCA.pem`.
   - Se hai installato il software AWS IoT Greengrass Core con il provisioning manuale o del parco veicoli, il file di certificato Amazon root CA potrebbe esistere in*/greengrass/v2*`/AmazonRootCA1.pem`.

   Se il certificato Amazon root CA non esiste in queste sedi, controlla la `system.rootCaPath` proprietà */greengrass/v2*`/config/effectiveConfig.yaml` per trovarne la posizione.
2. Aggiungi il contenuto del file di certificato CA del server proxy al file di certificato CA root di Amazon.

   L'esempio seguente mostra un certificato CA del server proxy aggiunto al file di certificato CA root di Amazon.

   ```
   -----BEGIN CERTIFICATE-----
   MIIEFTCCAv2gAwIQWgIVAMHSAzWG/5YVRYtRQOxXUTEpHuEmApzGCSqGSIb3DQEK
   \nCwUAhuL9MQswCQwJVUzEPMAVUzEYMBYGA1UECgwP1hem9uLmNvbSBJbmMuMRww
   ... content of proxy CA certificate ...
   ```

```
+vHIRlt0e5JAm5\noTIZGoFbK82A0/nO7f/t5PSIDAim9V3Gc3pSXxCCAQoFYnui
GaPUlGk1gCE84a0X\n7Rp/lND/PuMZ/s8YjlkY2NmYmNjMCAXDTE5MTEyN2cM216
gJMIADggEPADf2/m45hzEXAMPLE=
-----END CERTIFICATE-----

-----BEGIN CERTIFICATE-----
MIIDQTCCAimgF6AwIBAgITBmyfz/5mjAo54vB4ikPmljZKyjANJmApzyMZFo6qBg
ADA5MQswCQYDVQQGEwJVUzEPMA0tMVT8QtPHRh8jrdkGA1UEChMGDV3QQDExBBKW
... content of root CA certificate ...
o/ufQJQWUCyziar1hem9uMRkwFwYVPSHCb2XV4cdFyQzR1KldZwgJcIQ6XUDgHaa
5MsI+yMRQ+hDaXJiobldXgjUka642M4UwtBV8oK2xJNDd2ZhwLnoQdeXeGADKkpy
rqXRfKoQnoZsG4q5WTP46EXAMPLE
-----END CERTIFICATE-----
```

L'oggetto NetworkProxy

Utilizza l'oggetto `networkProxy` per specificare le informazioni sul proxy di rete. Questo oggetto contiene le seguenti informazioni:

`noProxyAddresses`

(Facoltativo) Un elenco separato da virgole di indirizzi IP o nomi host che sono esenti dal proxy.

`proxy`

Il proxy a cui connettersi. Questo oggetto contiene le seguenti informazioni:

`url`

L'URL del server proxy nel formato`scheme://userinfo@host:port`.

- `scheme`— Lo schema, che deve essere `http` o`https`.

> **Important**
>
> I dispositivi core Greengrass devono eseguire Greengrass nucleus v2.5.0 o versione successiva per utilizzare i proxy HTTPS.
> Se configuri un proxy HTTPS, devi aggiungere il certificato CA del server proxy al certificato Amazon root CA del dispositivo principale. Per ulteriori informazioni, consulta Abilita il dispositivo principale in modo che consideri attendibile un proxy HTTPS.

- `userinfo`— (Facoltativo) Le informazioni sul nome utente e sulla password. Se si specificano queste informazioni nel`url`, il dispositivo principale Greengrass ignora i `username` campi and. `password`
- `host`— Il nome host o l'indirizzo IP del server proxy.
- `port`— (Facoltativo) Il numero di porta. Se non specifichi la porta, il dispositivo principale Greengrass utilizza i seguenti valori predefiniti:
  - `http`— 80
  - `https`— 443

`username`

(Facoltativo) Il nome utente che autentica il server proxy.

`password`

(Facoltativo) La password che autentica il server proxy.

Utilizza un certificato del dispositivo firmato da una CA privata

Se si utilizza un'autorità di certificazione (CA) privata personalizzata, è necessario impostare il «nucleo **`greengrassDataPlaneEndpoint`** Greengrass» su. **`iotdata`** È possibile impostare questa opzione durante la distribuzione o l'installazione utilizzando l'argomento **`--init-config`** installer.

È possibile personalizzare l'endpoint del piano dati Greengrass a cui si connette il dispositivo. È possibile impostare questa opzione di configurazione in **`iotdata`** modo da impostare l'endpoint del piano dati Greengrass sullo stesso endpoint dell'endpoint dati IoT, che è possibile specificare con. **`iotDataEndpoint`**

Configurare i timeout MQTT e le impostazioni della cache

Nell' AWS IoT Greengrass ambiente, i componenti possono utilizzare MQTT per comunicare con. AWS IoT Core Il software AWS IoT Greengrass Core gestisce i messaggi MQTT per i componenti. Quando il dispositivo principale perde la connessione a Cloud AWS, il software memorizza nella cache i messaggi MQTT per riprovare più tardi quando la connessione viene ripristinata. È possibile configurare impostazioni come i timeout dei messaggi e la dimensione della cache. Per ulteriori informazioni, vedere i parametri `mqtt` e i parametri di `mqtt.spooler` configurazione del componente Greengrass nucleus.

AWS IoT Core impone quote di servizio al suo broker di messaggi MQTT. Queste quote potrebbero applicarsi ai messaggi inviati tra dispositivi principali e. AWS IoT Core Per ulteriori informazioni, consulta le quote del servizio di mediazione AWS IoT Core messaggi nel. Riferimenti generali di AWS

Aggiornamento del software AWS IoT Greengrass Core (OTA)

Il software AWS IoT Greengrass Core comprende il componente Greengrass nucleus e altri componenti opzionali che è possibile distribuire sui dispositivi per eseguire aggiornamenti over-the-air (OTA) del software. Questa funzionalità è integrata nel software Core. AWS IoT Greengrass

Gli aggiornamenti OTA rendono più efficiente:

- la correzione delle vulnerabilità in termini di sicurezza;
- la risoluzione dei problemi di stabilità del software;
- la distribuzione delle funzionalità nuove o migliorate.

Argomenti

Requisiti

I seguenti requisiti si applicano alla distribuzione degli aggiornamenti OTA del software AWS IoT Greengrass Core:

- Il dispositivo principale Greengrass deve disporre di una connessione a per Cloud AWS ricevere la distribuzione.
- Il dispositivo principale Greengrass deve essere configurato correttamente e dotato di certificati e chiavi per l'autenticazione con e. AWS IoT Core AWS IoT Greengrass
- Il software AWS IoT Greengrass Core deve essere configurato e funzionante come servizio di sistema. Gli aggiornamenti OTA non funzionano se si esegue il nucleo dal file JAR,`Greengrass.jar`. Per ulteriori informazioni, consulta Configurare il nucleo Greengrass come servizio di sistema.

Considerazioni per i dispositivi principali

Prima di eseguire un aggiornamento OTA, tieni presente l'impatto sui dispositivi principali che aggiorni e sui relativi dispositivi client collegati:

- Il nucleo Greengrass si spegne.
- Inoltre, tutti i componenti in esecuzione sul dispositivo principale si sono spenti. Se tali componenti scrivono su risorse locali, potrebbero lasciare tali risorse in uno stato errato a meno che non vengano disattivati correttamente. I componenti possono utilizzare la comunicazione tra processi per dire al componente nucleus di posticipare l'aggiornamento fino a quando non ripuliscono le risorse che utilizzano.
- Quando il componente nucleus viene spento, il dispositivo principale perde le connessioni con i Cloud AWS dispositivi locali. Il dispositivo principale non indirizzerà i messaggi dai dispositivi client quando è spento.
- Le funzioni Lambda di lunga durata eseguite come componenti perdono le informazioni sullo stato dinamico e interrompono tutto il lavoro in sospeso.

Comportamento dell'aggiornamento del nucleo di Greengrass

Quando si distribuisce un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi.

Quando la versione del componente Greengrass nucleus cambia, il software AWS IoT Greengrass Core, che include il nucleo e tutti gli altri componenti del dispositivo, si riavvia per applicare le modifiche. A causa dell'impatto sui dispositivi principali quando il componente nucleus viene aggiornato, potresti voler controllare quando una nuova versione della patch nucleus viene implementata sui tuoi dispositivi. A tal fine, è necessario includere direttamente il componente Greengrass nucleus nella distribuzione. L'inclusione diretta di un componente significa includere una versione specifica di quel componente nella configurazione di distribuzione e non fare affidamento sulle dipendenze dei componenti per distribuire quel componente sui dispositivi. Per ulteriori informazioni sulla definizione delle dipendenze nelle ricette dei componenti, consulta. Formato della ricetta

Consulta la tabella seguente per comprendere il comportamento di aggiornamento del componente Greengrass nucleus in base alle azioni e alle configurazioni di distribuzione.

Azione	Configurazione dell'implementazione	Comportamento di aggiornamento di Nucleus
Aggiungi nuovi dispositivi a un gruppo di oggetti destinato a una distribuzione esistente senza modificare la distribuzione.	L'implementazione non include direttamente Greengrass nucleus. La distribuzione include direttamente almeno un componente AWS fornito o include un componente personalizzato che dipende da un componente AWS fornito o dal nucleo Greengrass.	Sui nuovi dispositivi, installa la versione patch più recente di nucleus che soddisfa tutti i requisiti di dipendenza dei componenti. Sui dispositivi esistenti, non aggiorna la versione installata del nucleus.
Aggiungi nuovi dispositivi a un gruppo di oggetti destinato a una distribuzione esistente senza modificare la distribuzione.	L'implementazione include direttamente una versione specifica del nucleo Greengrass.	Sui nuovi dispositivi, installa la versione del nucleo specificata. Sui dispositivi esistenti, non aggiorna la versione installata del nucleus.
Crea una nuova distribuzione o modifica una distribuzione esistente.	L'implementazione non include direttamente Greengrass nucleus. La distribuzione include direttamente almeno un componente AWS fornito o include un componente personalizzato che dipende da un componente AWS fornito o dal nucleo Greengrass.	Su tutti i dispositivi interessati, installa la versione patch più recente del nucleus che soddisfa tutti i requisiti di dipendenza dai componenti, inclusi tutti i nuovi dispositivi aggiunti al gruppo di oggetti di destinazione.

Azione	Configurazione dell'implementazione	Comportamento di aggiornamento di Nucleus
Crea una nuova distribuzione o modifica una distribuzione esistente.	L'implementazione include direttamente una versione specifica del nucleo Greengrass.	Su tutti i dispositivi di destinazione, installa la versione di nucleus specificata, inclusi tutti i nuovi dispositivi aggiunti al gruppo di oggetti di destinazione.

Esegui un aggiornamento OTA

Per eseguire un aggiornamento OTA, crea una distribuzione che includa il componente nucleus e la versione da installare.

Disinstalla il software AWS IoT Greengrass Core

Puoi disinstallare il software AWS IoT Greengrass Core per rimuoverlo da un dispositivo che non desideri utilizzare come dispositivo principale Greengrass. Puoi anche utilizzare questi passaggi per ripulire un'installazione che non va a buon fine.

Per disinstallare il software AWS IoT Greengrass Core

1. Se si esegue il software come servizio di sistema, è necessario interrompere, disabilitare e rimuovere il servizio. Eseguite i seguenti comandi in base al sistema operativo in uso.

   Linux

   1. Arresta il servizio .

      ```
      sudo systemctl stop greengrass.service
      ```

   2. Disabilita il servizio.

      ```
      sudo systemctl disable greengrass.service
      ```

   3. Rimuovi il servizio.

```
sudo rm /etc/systemd/system/greengrass.service
```

4. Verifica che il servizio sia stato eliminato.

```
sudo systemctl daemon-reload && sudo systemctl reset-failed
```

Windows (Command Prompt)

> **Note**
>
> È necessario eseguire il prompt dei comandi come amministratore per eseguire questi comandi.

1. Arresta il servizio .

```
sc stop "greengrass"
```

2. Disabilita il servizio.

```
sc config "greengrass" start=disabled
```

3. Rimuovi il servizio.

```
sc delete "greengrass"
```

4. Riavviare il dispositivo.

Windows (PowerShell)

> **Note**
>
> È necessario eseguire PowerShell l'esecuzione come amministratore per eseguire questi comandi.

1. Arresta il servizio .

```
Stop-Service -Name "greengrass"
```

2. Disabilita il servizio.

```
Set-Service -Name "greengrass" -Status stopped -StartupType disabled
```

3. Rimuovi il servizio.

   - Per PowerShell 6.0 e versioni successive:

```
Remove-Service -Name "greengrass" -Confirm:$false -Verbose
```

   - Per PowerShell le versioni precedenti alla 6.0:

```
Get-Item HKLM:\SYSTEM\CurrentControlSet\Services\greengrass | Remove-Item
 -Force -Verbose
```

4. Riavviare il dispositivo.

2. Rimuovi la cartella principale dal dispositivo. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella principale.

   Linux

```
sudo rm -rf /greengrass/v2
```

   Windows (Command Prompt)

```
rmdir /s /q C:\greengrass\v2
```

   Windows (PowerShell)

```
cmd.exe /c "rmdir /s /q C:\greengrass\v2"
```

3. Elimina il dispositivo principale dal AWS IoT Greengrass servizio. Questo passaggio rimuove le informazioni sullo stato del dispositivo principale daCloud AWS. Assicurati di completare questo passaggio se prevedi di reinstallare il software AWS IoT Greengrass Core su un dispositivo principale con lo stesso nome.

   - Per eliminare un dispositivo principale dalla AWS IoT Greengrass console, procedi come segue:

a. Passare alla console AWS IoT Greengrass.
b. Scegli Dispositivi principali.
c. Scegli il dispositivo principale da eliminare.
d. Scegli Elimina.
e. Nella modalità di conferma, scegli Elimina.

- Per eliminare un dispositivo principale conAWS Command Line Interface, utilizzate l'DeleteCoreDeviceoperazione. Eseguite il comando seguente e *MyGreengrassCore*sostituitelo con il nome del dispositivo principale.

```
aws greengrassv2 delete-core-device --core-device-thing-name MyGreengrassCore
```

Tutorial di AWS IoT Greengrass V2

Puoi completare i seguenti tutorial per saperne di più sulle AWS IoT Greengrass V2 sue funzionalità.

Argomenti

Tutorial: Sviluppa un componente Greengrass che rinvii gli aggiornamenti dei componenti

Puoi completare questo tutorial per sviluppare un componente che differisca gli aggiornamenti della over-the-air distribuzione. Quando distribuisci aggiornamenti sui tuoi dispositivi, potresti voler ritardare gli aggiornamenti in base a condizioni, come le seguenti:

- Il dispositivo ha un livello di batteria basso.
- Il dispositivo sta eseguendo un processo o un processo che non può essere interrotto.
- Il dispositivo dispone di una connessione Internet limitata o costosa.

> **Note**
>
> Un componente è un modulo software che viene eseguito sui dispositivi AWS IoT Greengrass principali. I componenti consentono di creare e gestire applicazioni complesse come elementi costitutivi discreti che puoi riutilizzare da un dispositivo core Greengrass a un altro.

In questo tutorial, esegui quanto indicato di seguito:

1. Installa il Greengrass Development Kit CLI (GDK CLI) sul tuo computer di sviluppo. La CLI di GDK fornisce funzionalità che aiutano a sviluppare componenti Greengrass personalizzati.
2. Sviluppa un componente Hello World che rinvii gli aggiornamenti dei componenti quando il livello della batteria del dispositivo principale è inferiore a una soglia. Questo componente sottoscrive le notifiche di aggiornamento utilizzando l'operazione SubscribeToComponentUpdatesIPC. Quando riceve la notifica, verifica se il livello della batteria è inferiore a una soglia personalizzabile. Se il livello della batteria è inferiore alla soglia, posticipa l'aggiornamento di 30 secondi utilizzando l'operazione DeferComponentUpdateIPC. Sviluppi questo componente sul tuo computer di sviluppo utilizzando la CLI GDK.

   > **Note**
   >
   > Questo componente legge il livello della batteria da un file creato sul dispositivo principale per imitare una batteria reale, quindi puoi completare questo tutorial su un dispositivo principale senza batteria.

3. Pubblica quel componente nel AWS IoT Greengrass servizio.
4. Distribuisci quel componente Cloud AWS da un dispositivo principale Greengrass per testarlo. Quindi, modifichi il livello virtuale della batteria sul dispositivo principale e crei implementazioni aggiuntive per vedere in che modo il dispositivo principale posticipa gli aggiornamenti quando il livello della batteria è basso.

Puoi aspettarti di dedicare 20-30 minuti a questo tutorial.

Prerequisiti

Per completare questo tutorial, è necessario quanto segue:

- Un Account AWS. Se non lo hai, consultare Configura un Account AWS.
- Un utente AWS Identity and Access Management (IAM) con autorizzazioni di amministratore.
- Un dispositivo core Greengrass con una connessione Internet. Per ulteriori informazioni su come configurare un dispositivo principale, vedereConfigurazione dei dispositivi AWS IoT Greengrass principali.
  - Python 3.6 o successivo installato per tutti gli utenti sul dispositivo principale e aggiunto alla `PATH` variabile di ambiente. Su Windows, è inoltre necessario che Python Launcher per Windows sia installato per tutti gli utenti.

> **Important**
>
> In Windows, Python non viene installato per tutti gli utenti per impostazione predefinita. Quando installi Python, devi personalizzare l'installazione per configurarla affinché il software AWS IoT Greengrass Core esegua gli script Python. Ad esempio, se usi il programma di installazione grafico di Python, procedi come segue:
>
> 1. Seleziona Installa il programma di avvio per tutti gli utenti (consigliato).
> 2. Scegli Customize installation.
> 3. Scegli Next.
> 4. Seleziona Install for all users.
> 5. Seleziona Add Python to environment variables.
> 6. Scegli Installa.
>
> Per ulteriori informazioni, consulta Usare Python su Windows nella documentazione di Python 3.

- Un computer di sviluppo simile a Windows, macOS o UNIX con una connessione Internet.
  - Python 3.6 o versione successiva installato sul computer di sviluppo.
  - Git installato sul tuo computer di sviluppo.
  - AWS Command Line Interface(AWS CLI) installato e configurato con credenziali sul computer di sviluppo. Per ulteriori informazioni, vedere Installazione, aggiornamento e disinstallazione AWS CLI e configurazione di AWS CLI nella Guida per l'utente. AWS Command Line Interface

> **Note**
>
> Se utilizzi un Raspberry Pi o un altro dispositivo ARM a 32 bit, installa V1. AWS CLI AWS CLI La versione 2 non è disponibile per i dispositivi ARM a 32 bit. Per ulteriori informazioni, vedere Installazione, aggiornamento e disinstallazione della AWS CLI versione 1.

Fase 1: Installazione della CLI del Greengrass Development Kit

Il Greengrass Development Kit CLI (GDK CLI) fornisce funzionalità che aiutano a sviluppare componenti Greengrass personalizzati. Puoi usare la CLI GDK per creare, creare e pubblicare componenti personalizzati.

Se non hai installato la CLI GDK sul tuo computer di sviluppo, completa i seguenti passaggi per installarla.

Per installare l'ultima versione della CLI GDK

1. Sul tuo computer di sviluppo, esegui il comando seguente per installare l'ultima versione della CLI GDK dal GitHub suo repository.

   ```
   python3 -m pip install -U git+https://github.com/aws-greengrass/aws-greengrass-gdk-cli.git@v1.6.2
   ```

2. Esegui il comando seguente per verificare che la CLI GDK sia stata installata correttamente.

   ```
   gdk --help
   ```

   Se il `gdk` comando non viene trovato, aggiungi la sua cartella a PATH.

   - Sui dispositivi Linux, aggiungilo `/home/`*`MyUser`*`/.local/bin` a PATH e sostituiscilo *`MyUser`*con il nome dell'utente.
   - Sui dispositivi Windows, aggiungi *`PythonPath`*`\\Scripts` a PATH e sostituisci *`PythonPath`*con il percorso della cartella Python sul tuo dispositivo.

Fase 2: Sviluppare un componente che differisca gli aggiornamenti

In questa sezione, sviluppi un componente Hello World in Python che posticipa gli aggiornamenti dei componenti quando il livello della batteria del dispositivo principale è inferiore a una soglia configurata quando distribuisci il componente. In questo componente, si utilizza l'interfaccia di comunicazione tra processi (IPC) nella versione 2 SDK per dispositivi AWS IoT per Python. Si utilizza l'operazione SubscribeToComponentUpdatesIPC per ricevere notifiche quando il dispositivo principale riceve una distribuzione. Quindi, si utilizza l'operazione DeferComponentUpdateIPC per posticipare o confermare l'aggiornamento in base al livello della batteria del dispositivo.

Sviluppare un componente Hello World che posticipi gli aggiornamenti

1. Sul tuo computer di sviluppo, crea una cartella per il codice sorgente del componente.

```
mkdir com.example.BatteryAwareHelloWorld
cd com.example.BatteryAwareHelloWorld
```

2. Utilizzate un editor di testo per creare un file denominato`gdk-config.json`. La CLI GDK legge dal file di configurazione GDK CLI, `gdk-config.json` denominato, per creare e pubblicare componenti. Questo file di configurazione esiste nella radice della cartella del componente.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

```
nano gdk-config.json
```

   Copiate il seguente codice JSON nel file.

   - Sostituisci *`Amazon`* con il tuo nome.
   - Sostituisci *`us-west-2`* con il Regione AWS luogo in cui funziona il tuo dispositivo principale. La CLI GDK pubblica il componente in questo. Regione AWS
   - *`greengrass-component-artifacts`*Sostituiscilo con il prefisso del bucket S3 da usare. Quando usi la CLI di GDK per pubblicare il componente, la CLI di GDK carica gli artefatti del componente nel bucket S3 il cui nome è formato da questo valore, Regione AWS dal e dal tuo ID utilizzando il seguente formato:. Account AWS *`bucketPrefix`*`-`*`region`*`-`*`accountId`*

     Ad esempio, se specifichi **`greengrass-component-artifacts`** and e **`us-west-2`** il tuo Account AWS ID è**`123456789012`**, la CLI di GDK utilizza il bucket S3 denominato. `greengrass-component-artifacts-us-west-2-123456789012`

```
{
  "component": {
    "com.example.BatteryAwareHelloWorld": {
      "author": "Amazon",
      "version": "NEXT_PATCH",
      "build": {
        "build_system" : "zip"
      },
      "publish": {
```

```
        "region": "us-west-2",
        "bucket": "greengrass-component-artifacts"
      }
    }
  },
  "gdk_version": "1.0.0"
}
```

Il file di configurazione specifica quanto segue:

- La versione da utilizzare quando la CLI di GDK pubblica il componente Greengrass nel servizio cloud. AWS IoT Greengrass `NEXT_PATCH`specifica di scegliere la versione della patch successiva all'ultima versione disponibile nel servizio cloud. AWS IoT Greengrass Se il componente non ha ancora una versione nel servizio AWS IoT Greengrass cloud, utilizza la `1.0.0` CLI di GDK.
- Il sistema di compilazione del componente. Quando si utilizza il sistema di `zip` compilazione, la CLI GDK impacchetta il codice sorgente del componente in un file ZIP che diventa il singolo artefatto del componente.
- Il Regione AWS punto in cui la CLI di GDK pubblica il componente Greengrass.
- Il prefisso per il bucket S3 in cui la CLI di GDK carica gli artefatti del componente.

3. Usa un editor di testo per creare il codice sorgente del componente in un file denominato. `main.py`

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

   ```
   nano main.py
   ```

   Copia il seguente codice Python nel file.

   ```
   import json
   import os
   import sys
   import time
   import traceback

   from pathlib import Path

   from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2
   ```

```
HELLO_WORLD_PRINT_INTERVAL = 15  # Seconds
DEFER_COMPONENT_UPDATE_INTERVAL = 30 * 1000  # Milliseconds


class BatteryAwareHelloWorldPrinter():
    def __init__(self, ipc_client: GreengrassCoreIPCClientV2, battery_file_path:
 Path, battery_threshold: float):
        self.battery_file_path = battery_file_path
        self.battery_threshold = battery_threshold
        self.ipc_client = ipc_client
        self.subscription_operation = None

    def on_component_update_event(self, event):
        try:
            if event.pre_update_event is not None:
                if self.is_battery_below_threshold():
                    self.defer_update(event.pre_update_event.deployment_id)
                    print('Deferred update for deployment %s' %
                          event.pre_update_event.deployment_id)
                else:
                    self.acknowledge_update(
                        event.pre_update_event.deployment_id)
                    print('Acknowledged update for deployment %s' %
                          event.pre_update_event.deployment_id)
            elif event.post_update_event is not None:
                print('Applied update for deployment')
        except:
            traceback.print_exc()

    def subscribe_to_component_updates(self):
        if self.subscription_operation == None:
            # SubscribeToComponentUpdates returns a tuple with the response and the
 operation.
            _, self.subscription_operation =
 self.ipc_client.subscribe_to_component_updates(
                on_stream_event=self.on_component_update_event)

    def close_subscription(self):
        if self.subscription_operation is not None:
            self.subscription_operation.close()
            self.subscription_operation = None

    def defer_update(self, deployment_id):
```

```
        self.ipc_client.defer_component_update(
            deployment_id=deployment_id,
 recheck_after_ms=DEFER_COMPONENT_UPDATE_INTERVAL)

    def acknowledge_update(self, deployment_id):
        # Specify recheck_after_ms=0 to acknowledge a component update.
        self.ipc_client.defer_component_update(
            deployment_id=deployment_id, recheck_after_ms=0)

    def is_battery_below_threshold(self):
        return self.get_battery_level() < self.battery_threshold

    def get_battery_level(self):
        # Read the battery level from the virtual battery level file.
        with self.battery_file_path.open('r') as f:
            data = json.load(f)
            return float(data['battery_level'])

    def print_message(self):
        message = 'Hello, World!'
        if self.is_battery_below_threshold():
            message += ' Battery level (%d) is below threshold (%d), so the
 component will defer updates' % (
                self.get_battery_level(), self.battery_threshold)
        else:
            message += ' Battery level (%d) is above threshold (%d), so the
 component will acknowledge updates' % (
                self.get_battery_level(), self.battery_threshold)
        print(message)


def main():
    # Read the battery threshold and virtual battery file path from command-line
 args.
    args = sys.argv[1:]
    battery_threshold = float(args[0])
    battery_file_path = Path(args[1])
    print('Reading battery level from %s and deferring updates when below %d' % (
        str(battery_file_path), battery_threshold))

    try:
        # Create an IPC client and a Hello World printer that defers component
 updates.
        ipc_client = GreengrassCoreIPCClientV2()
```

```
        hello_world_printer = BatteryAwareHelloWorldPrinter(
            ipc_client, battery_file_path, battery_threshold)
        hello_world_printer.subscribe_to_component_updates()
        try:
            # Keep the main thread alive, or the process will exit.
            while True:
                hello_world_printer.print_message()
                time.sleep(HELLO_WORLD_PRINT_INTERVAL)
        except InterruptedError:
            print('Subscription interrupted')
        hello_world_printer.close_subscription()
    except Exception:
        print('Exception occurred', file=sys.stderr)
        traceback.print_exc()
        exit(1)


if __name__ == '__main__':
    main()
```

Questa applicazione Python esegue le seguenti operazioni:

- Legge il livello della batteria del dispositivo principale da un file virtuale del livello di batteria che creerai successivamente sul dispositivo principale. Questo file virtuale del livello della batteria imita una batteria reale, quindi puoi completare questo tutorial sui dispositivi principali che non dispongono di batteria.
- Legge gli argomenti della riga di comando per la soglia della batteria e il percorso del file virtuale del livello della batteria. La ricetta del componente imposta questi argomenti della riga di comando in base ai parametri di configurazione, in modo da poter personalizzare questi valori quando si distribuisce il componente.
- Utilizza il client IPC V2 nella versione 2 SDK per dispositivi AWS IoTper Python per comunicare con il software Core. AWS IoT Greengrass Rispetto al client IPC originale, il client IPC V2 riduce la quantità di codice da scrivere per utilizzare IPC nei componenti personalizzati.
- Si iscrive alle notifiche di aggiornamento utilizzando l'operazione IPC. SubscribeToComponentUpdates Il software AWS IoT Greengrass Core invia notifiche prima e dopo ogni implementazione. Il componente richiama la seguente funzione ogni volta che riceve una notifica. Se la notifica riguarda una distribuzione imminente, il componente verifica se il livello della batteria è inferiore a una soglia. Se il livello della batteria è inferiore

alla soglia, il componente posticipa l'aggiornamento di 30 secondi utilizzando l'operazione DeferComponentUpdateIPC. Altrimenti, se il livello della batteria non è inferiore alla soglia, il componente riconosce l'aggiornamento, quindi l'aggiornamento può procedere.

```
def on_component_update_event(self, event):
    try:
        if event.pre_update_event is not None:
            if self.is_battery_below_threshold():
                self.defer_update(event.pre_update_event.deployment_id)
                print('Deferred update for deployment %s' %
                      event.pre_update_event.deployment_id)
            else:
                self.acknowledge_update(
                    event.pre_update_event.deployment_id)
                print('Acknowledged update for deployment %s' %
                      event.pre_update_event.deployment_id)
        elif event.post_update_event is not None:
            print('Applied update for deployment')
    except:
        traceback.print_exc()
```

> **Note**
>
> Il software AWS IoT Greengrass Core non invia notifiche di aggiornamento per le distribuzioni locali, quindi distribuisci questo componente utilizzando il servizio AWS IoT Greengrass cloud per testarlo.

4. Utilizza un editor di testo per creare la ricetta del componente in un file denominato o. `recipe.json` `recipe.yaml` La ricetta del componente definisce i metadati del componente, i parametri di configurazione predefiniti e gli script del ciclo di vita specifici della piattaforma.

   JSON

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

   ```
   nano recipe.json
   ```

   Copiate il seguente codice JSON nel file.

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "COMPONENT_NAME",
  "ComponentVersion": "COMPONENT_VERSION",
  "ComponentDescription": "This Hello World component defers updates when the
 battery level is below a threshold.",
  "ComponentPublisher": "COMPONENT_AUTHOR",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "BatteryThreshold": 50,
      "LinuxBatteryFilePath": "/home/ggc_user/virtual_battery.json",
      "WindowsBatteryFilePath": "C:\\Users\\ggc_user\\virtual_battery.json"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk --upgrade",
        "run": "python3 -u {artifacts:decompressedPath}/
com.example.BatteryAwareHelloWorld/main.py \"{configuration:/BatteryThreshold}\"
 \"{configuration:/LinuxBatteryFilePath}\""
      },
      "Artifacts": [
        {
          "Uri": "s3://BUCKET_NAME/COMPONENT_NAME/COMPONENT_VERSION/
com.example.BatteryAwareHelloWorld.zip",
          "Unarchive": "ZIP"
        }
      ]
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk --upgrade",
        "run": "py -3 -u {artifacts:decompressedPath}/
com.example.BatteryAwareHelloWorld/main.py \"{configuration:/BatteryThreshold}\"
 \"{configuration:/WindowsBatteryFilePath}\""
      },
```

```
      "Artifacts": [
        {
          "Uri": "s3://BUCKET_NAME/COMPONENT_NAME/COMPONENT_VERSION/
com.example.BatteryAwareHelloWorld.zip",
          "Unarchive": "ZIP"
        }
      ]
    }
  ]
}
```

YAML

Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

```
nano recipe.yaml
```

Copiate il seguente codice YAML nel file.

```
---
RecipeFormatVersion: "2020-01-25"
ComponentName: "COMPONENT_NAME"
ComponentVersion: "COMPONENT_VERSION"
ComponentDescription: "This Hello World component defers updates when the
 battery level is below a threshold."
ComponentPublisher: "COMPONENT_AUTHOR"
ComponentConfiguration:
  DefaultConfiguration:
    BatteryThreshold: 50
    LinuxBatteryFilePath: "/home/ggc_user/virtual_battery.json"
    WindowsBatteryFilePath: "C:\\Users\\ggc_user\\virtual_battery.json"
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: python3 -m pip install --user awsiotsdk --upgrade
      run: python3 -u {artifacts:decompressedPath}/
com.example.BatteryAwareHelloWorld/main.py "{configuration:/BatteryThreshold}"
 "{configuration:/LinuxBatteryFilePath}"
    Artifacts:
```

```
      - Uri: "s3://BUCKET_NAME/COMPONENT_NAME/COMPONENT_VERSION/
com.example.BatteryAwareHelloWorld.zip"
        Unarchive: ZIP
  - Platform:
      os: windows
    Lifecycle:
      install: py -3 -m pip install --user awsiotsdk --upgrade
      run: py -3 -u {artifacts:decompressedPath}/
com.example.BatteryAwareHelloWorld/main.py "{configuration:/BatteryThreshold}"
 "{configuration:/WindowsBatteryFilePath}"
    Artifacts:
      - Uri: "s3://BUCKET_NAME/COMPONENT_NAME/COMPONENT_VERSION/
com.example.BatteryAwareHelloWorld.zip"
        Unarchive: ZIP
```

Questa ricetta specifica quanto segue:

- Parametri di configurazione predefiniti per la soglia della batteria, il percorso del file della batteria virtuale sui dispositivi core Linux e il percorso del file della batteria virtuale sui dispositivi core Windows.
- Un `install` ciclo di vita che installa l'ultima versione della SDK per dispositivi AWS IoT v2 per Python.
- Un `run` ciclo di vita in cui viene eseguita l'applicazione Python. `main.py`
- Segnaposti, come `COMPONENT_NAME` e`COMPONENT_VERSION`, dove la CLI GDK sostituisce le informazioni quando crea la ricetta del componente.

Per ulteriori informazioni sulle ricette dei componenti, consulta. AWS IoT Greengrassriferimento alla ricetta del componente

Fase 3: Pubblicare il componente nel AWS IoT Greengrass servizio

In questa sezione, pubblichi il componente Hello World nel servizio AWS IoT Greengrass cloud. Dopo che un componente è disponibile nel servizio AWS IoT Greengrass cloud, puoi distribuirlo sui dispositivi principali. Utilizzi la CLI GDK per pubblicare il componente dal tuo computer di sviluppo AWS IoT Greengrass al servizio cloud. La CLI di GDK carica la ricetta e gli artefatti del componente per te.

Per pubblicare il componente Hello World nel servizio AWS IoT Greengrass

1. Esegui il comando seguente per creare il componente utilizzando la CLI GDK. Il comando component build crea una ricetta e artefatti basati sul file di configurazione GDK CLI. In questo processo, la CLI GDK crea un file ZIP che contiene il codice sorgente del componente.

   ```
   gdk component build
   ```

   Dovreste vedere messaggi simili a quelli dell'esempio seguente.

   ```
   [2022-04-28 11:20:16] INFO - Getting project configuration from gdk-config.json
   [2022-04-28 11:20:16] INFO - Found component recipe file 'recipe.yaml' in the
    project directory.
   [2022-04-28 11:20:16] INFO - Building the component
    'com.example.BatteryAwareHelloWorld' with the given project configuration.
   [2022-04-28 11:20:16] INFO - Using 'zip' build system to build the component.
   [2022-04-28 11:20:16] WARNING - This component is identified as using 'zip' build
    system. If this is incorrect, please exit and specify custom build command in the
    'gdk-config.json'.
   [2022-04-28 11:20:16] INFO - Zipping source code files of the component.
   [2022-04-28 11:20:16] INFO - Copying over the build artifacts to the greengrass
    component artifacts build folder.
   [2022-04-28 11:20:16] INFO - Updating artifact URIs in the recipe.
   [2022-04-28 11:20:16] INFO - Creating component recipe in 'C:\Users\finthomp
   \greengrassv2\com.example.BatteryAwareHelloWorld\greengrass-build\recipes'.
   ```

2. Esegui il comando seguente per pubblicare il componente nel servizio AWS IoT Greengrass cloud. Il comando component publish carica l'elemento del file ZIP del componente in un bucket S3. Quindi, aggiorna l'URI S3 del file ZIP nella ricetta del componente e carica la ricetta sul servizio. AWS IoT Greengrass In questo processo, la CLI di GDK verifica quale versione del componente Hello World è già disponibile nel servizio cloud, in AWS IoT Greengrass modo da poter scegliere la versione della patch successiva a quella versione. Se il componente non esiste ancora, la CLI di GDK utilizza la versione. `1.0.0`

   ```
   gdk component publish
   ```

   Dovreste vedere messaggi simili all'esempio seguente. L'output indica la versione del componente creata dalla CLI GDK.

   ```
   [2022-04-28 11:20:29] INFO - Getting project configuration from gdk-config.json
   ```

```
[2022-04-28 11:20:29] INFO - Found component recipe file 'recipe.yaml' in the
 project directory.
[2022-04-28 11:20:29] INFO - Found credentials in shared credentials file: ~/.aws/
credentials
[2022-04-28 11:20:30] INFO - No private version of the component
 'com.example.BatteryAwareHelloWorld' exist in the account. Using '1.0.0' as the
 next version to create.
[2022-04-28 11:20:30] INFO - Publishing the component
 'com.example.BatteryAwareHelloWorld' with the given project configuration.
[2022-04-28 11:20:30] INFO - Uploading the component built artifacts to s3 bucket.
[2022-04-28 11:20:30] INFO - Uploading component artifacts to S3
 bucket: greengrass-component-artifacts-us-west-2-123456789012. If this is your
 first time using this bucket, add the 's3:GetObject' permission to each core
 device's token exchange role to allow it to download the component artifacts. For
 more information, see https://docs.aws.amazon.com/greengrass/v2/developerguide/
device-service-role.html.
[2022-04-28 11:20:30] INFO - Not creating an artifacts bucket as it already exists.
[2022-04-28 11:20:30] INFO - Updating the component recipe
 com.example.BatteryAwareHelloWorld-1.0.0.
[2022-04-28 11:20:31] INFO - Creating a new greengrass component
 com.example.BatteryAwareHelloWorld-1.0.0
[2022-04-28 11:20:31] INFO - Created private version '1.0.0' of the component in
 the account.'com.example.BatteryAwareHelloWorld'.
```

3. Copia il nome del bucket S3 dall'output. Utilizzerai il nome del bucket in un secondo momento per consentire al dispositivo principale di scaricare gli artefatti dei componenti da questo bucket.
4. (Facoltativo) Visualizza il componente nella AWS IoT Greengrass console per verificare che sia stato caricato correttamente. Esegui questa operazione:

   a. Nel menu di navigazione della AWS IoT Greengrassconsole, scegli Componenti.

   b. Nella pagina Componenti, scegli la scheda I miei componenti, quindi scegli com.example.BatteryAwareHelloWorld.

   In questa pagina, puoi vedere la ricetta del componente e altre informazioni sul componente.
5. Consenti al dispositivo principale di accedere agli artefatti dei componenti nel bucket S3.

   Ogni dispositivo principale ha un ruolo IAM del dispositivo principale che gli consente di interagire AWS IoT e inviare log al cloud. AWS Per impostazione predefinita, questo ruolo del dispositivo non consente l'accesso ai bucket S3, quindi è necessario creare e allegare una policy che consenta al dispositivo principale di recuperare gli artefatti dei componenti dal bucket S3.

Se il ruolo del tuo dispositivo consente già l'accesso al bucket S3, puoi saltare questo passaggio. Altrimenti, crea una policy IAM che consenta l'accesso e collegala al ruolo, come segue:

a. Nel menu di navigazione della console IAM, scegli Policies, quindi scegli Crea policy.

b. Nella scheda JSON, sostituire il contenuto del segnaposto con la seguente policy. Sostituisci *greengrass-component-artifacts-us-west-2-123456789012* con il nome del bucket S3 in cui la CLI GDK ha caricato gli artefatti del componente.

   Ad esempio, se hai specificato **greengrass-component-artifacts** and **us-west-2** nel file di configurazione della CLI di GDK e Account AWS il tuo ID **123456789012** è, la CLI di GDK utilizza il bucket S3 denominato. `greengrass-component-artifacts-us-west-2-123456789012`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::greengrass-component-artifacts-us-
west-2-123456789012/*"
    }
  ]
}
```

c. Seleziona Avanti.

d. Nella sezione Dettagli della politica, per Nome, inserisci. **MyGreengrassV2ComponentArtifactPolicy**

e. Scegli Crea policy.

f. Nel menu di navigazione della console IAM, scegli Ruolo, quindi scegli il nome del ruolo per il dispositivo principale. Hai specificato questo nome di ruolo quando hai installato il software AWS IoT Greengrass Core. Se non hai specificato un nome, l'impostazione predefinita è`GreengrassV2TokenExchangeRole`.

g. In Autorizzazioni, scegli Aggiungi autorizzazioni, quindi scegli Allega politiche.

h. Nella pagina Aggiungi autorizzazioni, seleziona la casella di controllo accanto alla **`MyGreengrassV2ComponentArtifactPolicy`** politica che hai creato, quindi scegli Aggiungi autorizzazioni.

Fase 4: Implementazione e test del componente su un dispositivo principale

In questa sezione, distribuisci il componente sul dispositivo principale per testarne la funzionalità. Sul dispositivo principale, crei il file virtuale del livello della batteria per imitare una batteria reale. Quindi, crei distribuzioni aggiuntive e osservi i file di registro dei componenti sul dispositivo principale per verificare che il componente rinvii e confermi gli aggiornamenti.

Per distribuire e testare il componente Hello World che posticipa gli aggiornamenti

1. Usa un editor di testo per creare un file virtuale del livello della batteria. Questo file imita una batteria reale.

   - Sui dispositivi principali Linux, crea un file denominato`/home/ggc_user/virtual_battery.json`. Esegui l'editor di testo con `sudo` autorizzazioni.
   - Sui dispositivi Windows principali, crea un file denominato`C:\Users\ggc_user\virtual_battery.json`. Esegui l'editor di testo come amministratore.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

   ```
   sudo nano /home/ggc_user/virtual_battery.json
   ```

   Copiate il seguente codice JSON nel file.

   ```
   {
     "battery_level": 50
   }
   ```

2. Distribuisci il componente Hello World sul dispositivo principale. Esegui questa operazione:

   a. Nel menu di navigazione AWS IoT Greengrassdella console, scegli Componenti.

   b. Nella pagina Componenti, scegli la scheda I miei componenti, quindi scegli com.example.BatteryAwareHelloWorld.

c. Nella pagina com.example.BatteryAwareHelloWorld, scegli (Distribuisci).

d. Da Aggiungi alla distribuzione, scegli una distribuzione esistente da modificare oppure scegli di creare una nuova distribuzione, quindi scegli Avanti.

e. Se hai scelto di creare una nuova distribuzione, scegli il dispositivo principale o il gruppo di oggetti di destinazione per la distribuzione. Nella pagina Specificare la destinazione, in Obiettivo di distribuzione, scegli un dispositivo principale o un gruppo di oggetti, quindi scegli Avanti.

f. Nella pagina Seleziona componenti, verifica che il com.example.BatteryAwareHelloWorldcomponente sia selezionato, scegli Avanti.

g. Nella pagina Configura componenti com.example.BatteryAwareHelloWorld, selezionate e quindi effettuate le seguenti operazioni:

   i. Scegli Configura componente.

   ii. Nella com.example.BatteryAwareHelloWorld modalità Configura, in Aggiornamento della configurazione, in Configurazione da unire, inserisci il seguente aggiornamento di configurazione.

   ```
   {
     "BatteryThreshold": 70
   }
   ```

   iii. Scegli Conferma per chiudere la modalità, quindi scegli Avanti.

h. Nella pagina Conferma impostazioni avanzate, nella sezione Criteri di distribuzione, in Politica di aggiornamento dei componenti, conferma che sia selezionata l'opzione Notifica componenti. L'opzione Notifica componenti è selezionata per impostazione predefinita quando si crea una nuova distribuzione.

i. Nella pagina Review (Verifica), scegli Deploy (Distribuisci).

   Il completamento della distribuzione può richiedere fino a un minuto.

3. Il software AWS IoT Greengrass Core salva lo stdout dai processi dei componenti nei file di registro nella `logs` cartella. Eseguite il comando seguente per verificare che il componente Hello World venga eseguito e stampi i messaggi di stato.

   Linux or Unix

   ```
   sudo tail -f /greengrass/v2/logs/com.example.BatteryAwareHelloWorld.log
   ```

Windows Command Prompt (CMD)

```
type C:\greengrass\v2\logs\com.example.BatteryAwareHelloWorld.log
```

PowerShell

```
gc C:\greengrass\v2\logs\com.example.BatteryAwareHelloWorld.log -Tail 10 -Wait
```

Dovreste vedere messaggi simili a quelli dell'esempio seguente.

```
Hello, World! Battery level (50) is below threshold (70), so the component will
 defer updates.
```

Note

Se il file non esiste, la distribuzione potrebbe non essere ancora completa. Se il file non esiste entro 30 secondi, è probabile che la distribuzione non sia riuscita. Ciò può verificarsi, ad esempio, se il dispositivo principale non è autorizzato a scaricare gli elementi del componente dal bucket S3. Esegui il comando seguente per visualizzare il file di registro del software AWS IoT Greengrass Core. Questo file include i log del servizio di distribuzione del dispositivo principale Greengrass.

Linux or Unix

```
sudo tail -f /greengrass/v2/logs/greengrass.log
```

Windows Command Prompt (CMD)

```
type C:\greengrass\v2\logs\greengrass.log
```

Il `type` comando scrive il contenuto del file nel terminale. Esegui questo comando più volte per osservare le modifiche nel file.

PowerShell

```
gc C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
```

4. Crea una nuova distribuzione sul dispositivo principale per verificare che il componente rinvii l'aggiornamento. Esegui questa operazione:

   a. Nel menu di navigazione AWS IoT Greengrassdella console, scegli Distribuzioni.

   b. Scegli la distribuzione che hai creato o modificato in precedenza.

   c. Nella pagina di distribuzione, scegli Rivedi.

   d. Nella modalità Revise deployment, scegli Revise deployment.

   e. Scegli Avanti in ogni passaggio, quindi scegli Distribuisci.

5. Esegui il comando seguente per visualizzare nuovamente i log del componente e verifica che posticipi l'aggiornamento.

   Linux or Unix

   ```
   sudo tail -f /greengrass/v2/logs/com.example.BatteryAwareHelloWorld.log
   ```

   Windows Command Prompt (CMD)

   ```
   type C:\greengrass\v2\logs\com.example.BatteryAwareHelloWorld.log
   ```

   PowerShell

   ```
   gc C:\greengrass\v2\logs\com.example.BatteryAwareHelloWorld.log -Tail 10 -Wait
   ```

   Dovrebbero comparire messaggi simili a quelli dell'esempio seguente. Il componente posticipa l'aggiornamento di 30 secondi, quindi stampa ripetutamente questo messaggio.

   ```
   Deferred update for deployment 50722a95-a05f-4e2a-9414-da80103269aa.
   ```

6. Utilizza un editor di testo per modificare il file virtuale del livello della batteria e imposta il livello della batteria a un valore superiore alla soglia, in modo che l'installazione possa procedere.

   - Sui dispositivi principali Linux, modifica il file denominato`/home/ggc_user/virtual_battery.json`. Esegui l'editor di testo con `sudo` le autorizzazioni.
   - Sui dispositivi Windows Core, modifica il file denominato`C:\Users\ggc_user\virtual_battery.json`. Esegui l'editor di testo come amministratore.

Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

```
sudo nano /home/ggc_user/virtual_battery.json
```

Cambia il livello della batteria in. 80

```
{
  "battery_level": 80
}
```

7. Esegui il comando seguente per visualizzare nuovamente i log del componente e verifica che riconosca l'aggiornamento.

   Linux or Unix

   ```
   sudo tail -f /greengrass/v2/logs/com.example.BatteryAwareHelloWorld.log
   ```

   Windows Command Prompt (CMD)

   ```
   type C:\greengrass\v2\logs\com.example.BatteryAwareHelloWorld.log
   ```

   PowerShell

   ```
   gc C:\greengrass\v2\logs\com.example.BatteryAwareHelloWorld.log -Tail 10 -Wait
   ```

   Dovreste vedere messaggi simili ai seguenti esempi.

   ```
   Hello, World! Battery level (80) is above threshold (70), so the component will
    acknowledge updates.
   Acknowledged update for deployment f9499eb2-4a40-40a7-86c1-c89887d859f1.
   ```

Hai completato questo tutorial. Il componente Hello World differisce o riconosce gli aggiornamenti in base al livello della batteria del dispositivo principale. Per ulteriori informazioni sugli argomenti trattati in questo tutorial, consulta quanto segue:

- Sviluppa AWS IoT Greengrass componenti
- Implementazione AWS IoT Greengrass dei componenti sui dispositivi
- Usa il SDK per dispositivi AWS IoT per comunicare con il nucleo Greengrass, altri componenti e AWS IoT Core
- AWS IoT GreengrassInterfaccia a riga di comando del Development Kit

Tutorial: Interagisci con i dispositivi IoT locali tramite MQTT

Puoi completare questo tutorial per configurare un dispositivo principale per interagire con i dispositivi IoT locali, chiamati dispositivi client, che si connettono al dispositivo principale tramite MQTT. In questo tutorial, configurerai AWS IoT le cose per utilizzare il cloud discovery per connetterti al dispositivo principale come dispositivi client. Quando configuri il cloud discovery, un dispositivo client può inviare una richiesta al servizio AWS IoT Greengrass cloud per scoprire i dispositivi principali. La risposta di AWS IoT Greengrass include informazioni sulla connettività e certificati per i dispositivi principali che configuri il dispositivo client per il rilevamento. Quindi, il dispositivo client può utilizzare queste informazioni per connettersi a un dispositivo principale disponibile in cui comunicare tramite MQTT.

In questo tutorial, esegui quanto indicato di seguito:

1. Rivedi e aggiorna le autorizzazioni del dispositivo principale, se necessario.
2. Associa i dispositivi client al dispositivo principale, in modo che possano scoprire il dispositivo principale utilizzando il cloud discovery.
3. Implementa i componenti Greengrass sul dispositivo principale per abilitare il supporto dei dispositivi client.
4. Connect i dispositivi client al dispositivo principale e testa la comunicazione con il servizio AWS IoT Core cloud.
5. Sviluppa un componente Greengrass personalizzato che comunichi con i dispositivi client.
6. Sviluppa un componente personalizzato che interagisca con le ombre dei dispositivi AWS IoT client.

Questo tutorial utilizza un singolo dispositivo core e un singolo dispositivo client. Puoi anche seguire il tutorial per connettere e testare più dispositivi client.

Puoi aspettarti di dedicare 30-60 minuti a questo tutorial.

Prerequisiti

Per completare questo tutorial, è necessario quanto segue:

- Un Account AWS. Se non lo hai, consultare Configura un Account AWS.
- Un utente AWS Identity and Access Management (IAM) con autorizzazioni di amministratore.
- Un dispositivo principale Greengrass. Per ulteriori informazioni su come configurare un dispositivo principale, vedereConfigurazione dei dispositivi AWS IoT Greengrass principali.
  - Il dispositivo principale deve eseguire Greengrass nucleus v2.6.0 o versione successiva. Questa versione include il supporto per le wildcard nelle comunicazioni locali di pubblicazione/sottoscrizione e il supporto per le ombre dei dispositivi client.

    > **Note**
    >
    > Il supporto per i dispositivi client richiede Greengrass nucleus v2.2.0 o versione successiva. Tuttavia, questo tutorial esplora le funzionalità più recenti, come il supporto per le wildcard MQTT nelle modalità di pubblicazione/sottoscrizione locali e il supporto per le ombre dei dispositivi client. Queste funzionalità richiedono Greengrass nucleus v2.6.0 o versione successiva.

  - Il dispositivo principale deve trovarsi sulla stessa rete dei dispositivi client per connettersi.
  - (Facoltativo) Per completare i moduli in cui si sviluppano componenti Greengrass personalizzati, il dispositivo principale deve eseguire la Greengrass CLI. Per ulteriori informazioni, consulta Installazione della CLI di Greengrass.
- Qualsiasi AWS IoT cosa da connettere come dispositivo client in questo tutorial. Per ulteriori informazioni, consulta AWS IoTCreate resources nella AWS IoT CoreDeveloper Guide.
  - La AWS IoT politica del dispositivo client deve consentire l'`greengrass:Discover`autorizzazione. Per ulteriori informazioni, consulta AWS IoTPolicy minima per i dispositivi client.
  - Il dispositivo client deve trovarsi sulla stessa rete del dispositivo principale.
  - Il dispositivo client deve eseguire Python 3.
  - Il dispositivo client deve eseguire Git.

Passaggio 1: rivedi e aggiorna la AWS IoT politica di base del dispositivo

Per supportare i dispositivi client, la AWS IoT politica di base del dispositivo deve consentire le seguenti autorizzazioni:

- `greengrass:PutCertificateAuthorities`
- `greengrass:VerifyClientDeviceIdentity`
- `greengrass:VerifyClientDeviceIoTCertificateAssociation`
- `greengrass:GetConnectivityInfo`
- `greengrass:UpdateConnectivityInfo`— (Facoltativo) Questa autorizzazione è necessaria per utilizzare il componente del rilevatore IP, che riporta le informazioni sulla connettività di rete del dispositivo principale al AWS IoT Greengrass servizio cloud.

Per ulteriori informazioni su queste autorizzazioni e AWS IoT politiche per i dispositivi principali, consulta Policy AWS IoT per operazioni del piano dei dati e. AWS IoTPolitica minima per supportare i dispositivi client

In questa sezione, esamini le AWS IoT politiche per il tuo dispositivo principale e aggiungi le autorizzazioni necessarie mancanti. Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per il provisioning delle risorse, il tuo dispositivo principale dispone di una AWS IoT politica che consente l'accesso a tutte le AWS IoT Greengrass azioni ()`greengrass:*`. In questo caso, è necessario aggiornare la AWS IoT policy solo se si prevede di configurare il componente shadow manager con cui sincronizzare le ombre del dispositivo. AWS IoT Core Altrimenti, puoi saltare questa sezione.

Per rivedere e aggiornare la politica di un dispositivo principale AWS IoT

1. Nel menu di navigazione della AWS IoT Greengrassconsole, scegli Dispositivi principali.
2. Nella pagina Dispositivi principali, scegli il dispositivo principale da aggiornare.
3. Nella pagina dei dettagli del dispositivo principale, scegli il link all'oggetto del dispositivo principale. Questo link apre la pagina dei dettagli dell'oggetto nella AWS IoT console.
4. Nella pagina dei dettagli dell'oggetto, scegli Certificati.
5. Nella scheda Certificati, scegli il certificato attivo dell'oggetto.
6. Nella pagina dei dettagli del certificato, scegli Politiche.
7. Nella scheda Politiche, scegli la AWS IoT politica da rivedere e aggiornare. Puoi aggiungere le autorizzazioni richieste a qualsiasi policy allegata al certificato attivo del dispositivo principale.

> **Note**
>
> Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per il provisioning delle risorse, hai due AWS IoT politiche. Ti consigliamo di scegliere la politica denominata GreengrassV2IoTThingPolicy, se esiste. I dispositivi principali creati con il programma di installazione rapida utilizzano questo nome di policy per impostazione predefinita. Se aggiungi autorizzazioni a questo criterio, concedi tali autorizzazioni anche ad altri dispositivi principali che utilizzano questo criterio.

8. Nella panoramica della politica, scegli Modifica versione attiva.
9. Rivedi la politica per le autorizzazioni richieste e aggiungi le autorizzazioni richieste mancanti.
10. Per impostare una nuova versione della politica come versione attiva, in Stato della versione della politica, seleziona Imposta la versione modificata come versione attiva per questa politica.
11. Scegli Salva come nuova versione.

Passaggio 2: abilitare il supporto per i dispositivi client

Affinché un dispositivo client utilizzi cloud discovery per connettersi a un dispositivo principale, è necessario associare i dispositivi. Quando associ un dispositivo client a un dispositivo principale, consenti a tale dispositivo client di recuperare gli indirizzi IP e i certificati del dispositivo principale da utilizzare per la connessione.

Per consentire ai dispositivi client di connettersi in modo sicuro a un dispositivo principale e comunicare con i componenti AWS IoT Core Greengrass, distribuisci i seguenti componenti Greengrass sul dispositivo principale:

- Autenticazione del dispositivo client (`aws.greengrass.clientdevices.Auth`)

  Implementate il componente di autenticazione dei dispositivi client per autenticare i dispositivi client e autorizzare le azioni dei dispositivi client. Questo componente consente ai tuoi dispositivi di AWS IoT connettersi a un dispositivo principale.

  Questo componente richiede alcune configurazioni per utilizzarlo. È necessario specificare i gruppi di dispositivi client e le operazioni che ciascun gruppo è autorizzato a eseguire, ad esempio la connessione e la comunicazione tramite MQTT. Per ulteriori informazioni, consultate Configurazione dei componenti di autenticazione dei dispositivi client.

- Broker MQTT 3.1.1 (Moquette) (`aws.greengrass.clientdevices.mqtt.Moquette`)

  Implementate il componente del broker MQTT Moquette per eseguire un broker MQTT leggero. Il broker Moquette MQTT è conforme a MQTT 3.1.1 e include il supporto locale per QoS 0, QoS 1, QoS 2, messaggi conservati, messaggi di ultima volontà e abbonamenti permanenti.

  Non è necessario configurare questo componente per utilizzarlo. Tuttavia, è possibile configurare la porta su cui questo componente gestisce il broker MQTT. Per impostazione predefinita, utilizza la porta 8883.
- Ponte MQTT (`aws.greengrass.clientdevices.mqtt.Bridge`)

  (Facoltativo) Implementate il componente bridge MQTT per inoltrare messaggi tra dispositivi client (MQTT locale), pubblicazione e sottoscrizione locali e MQTT. AWS IoT Core Configura questo componente per sincronizzare i dispositivi client AWS IoT Core e interagire con i dispositivi client dai componenti Greengrass.

  Questo componente richiede una configurazione per essere utilizzato. È necessario specificare le mappature degli argomenti in cui questo componente inoltra i messaggi. Per ulteriori informazioni, vedere Configurazione dei componenti del bridge MQTT.
- Rilevatore IP (`aws.greengrass.clientdevices.IPDetector`)

  (Facoltativo) Implementate il componente del rilevatore IP per segnalare automaticamente gli endpoint del broker MQTT del dispositivo principale al servizio cloud. AWS IoT Greengrass Non è possibile utilizzare questo componente se si dispone di una configurazione di rete complessa, ad esempio una in cui un router inoltra la porta del broker MQTT al dispositivo principale.

  Non è necessario configurare questo componente per utilizzarlo.

In questa sezione, si utilizza la AWS IoT Greengrass console per associare i dispositivi client e distribuire i componenti dei dispositivi client su un dispositivo principale.

Per abilitare il supporto dei dispositivi client

1. Passare alla console AWS IoT Greengrass.
2. Nel menu di navigazione a sinistra, scegli Dispositivi principali.
3. Nella pagina Dispositivi principali, scegli il dispositivo principale su cui desideri abilitare il supporto per i dispositivi client.
4. Nella pagina dei dettagli del dispositivo principale, scegli la scheda Dispositivi client.

5. Nella scheda Dispositivi client, scegli Configura cloud discovery.

   Viene visualizzata la pagina Configura il rilevamento dei dispositivi principali. In questa pagina è possibile associare i dispositivi client a un dispositivo principale e distribuire i componenti dei dispositivi client. Questa pagina seleziona il dispositivo principale per te nel Passaggio 1: Seleziona i dispositivi principali di destinazione.

   > **Note**
   >
   > È inoltre possibile utilizzare questa pagina per configurare l'individuazione dei dispositivi principali per un gruppo di oggetti. Se si sceglie questa opzione, è possibile distribuire i componenti dei dispositivi client su tutti i dispositivi principali di un gruppo di oggetti. Tuttavia, se si sceglie questa opzione, è necessario associare manualmente i dispositivi client a ciascun dispositivo principale in un secondo momento dopo aver creato la distribuzione. In questo tutorial, configurerai un dispositivo single core.

6. Nel passaggio 2: Associa i dispositivi client, associa l'AWS IoTelemento del dispositivo client al dispositivo principale. Ciò consente al dispositivo client di utilizzare il cloud discovery per recuperare le informazioni e i certificati di connettività del dispositivo principale. Esegui questa operazione:

   a. Scegli Associa dispositivi client.

   b. Nella modalità Associa i dispositivi client al dispositivo principale, inserisci il nome dell'AWS IoToggetto da associare.

   c. Scegli Aggiungi.

   d. Selezionare Associate (Associa).

7. Nel passaggio 3: Configurare e distribuire i componenti Greengrass, distribuire i componenti per abilitare il supporto dei dispositivi client. Se il dispositivo principale di destinazione ha una distribuzione precedente, questa pagina rivede tale distribuzione. Altrimenti, questa pagina crea una nuova distribuzione per il dispositivo principale. Effettua le seguenti operazioni per configurare e distribuire i componenti del dispositivo client:

   a. Il dispositivo principale deve eseguire Greengrass nucleus v2.6.0 o versione successiva per completare questo tutorial. Se il dispositivo principale esegue una versione precedente, procedi come segue:

      i. Seleziona la casella per distribuire il aws.greengrass.Nucleuscomponente.

ii. Per il aws.greengrass.Nucleuscomponente, scegli Modifica configurazione.

iii. Per la versione del componente, scegli la versione 2.6.0 o successiva.

iv. Scegli Conferma.

> **Note**
>
> Se si aggiorna il nucleo Greengrass da una versione secondaria precedente e il dispositivo principale esegue componenti AWS forniti che dipendono dal nucleo, è necessario aggiornare anche i componenti forniti alle AWS versioni più recenti. Puoi configurare la versione di questi componenti quando esaminerai la distribuzione più avanti in questo tutorial. Per ulteriori informazioni, consulta Aggiornamento del software AWS IoT Greengrass Core (OTA).

b. Per il aws.greengrass.clientdevices.Authcomponente, scegli Modifica configurazione.

c. Nella modalità Modifica configurazione per il componente di autenticazione del dispositivo client, configurate una politica di autorizzazione che consenta ai dispositivi client di pubblicare e sottoscrivere il broker MQTT sul dispositivo principale. Esegui questa operazione:

i. In Configurazione, nel blocco Configurazione per unire codice, immettete la seguente configurazione, che contiene una politica di autorizzazione del dispositivo client. Ogni politica di autorizzazione del gruppo di dispositivi specifica una serie di azioni e le risorse su cui un dispositivo client può eseguire tali azioni.

- Questa policy consente ai dispositivi client i cui nomi iniziano con di `MyClientDevice` connettersi e comunicare su tutti gli argomenti MQTT. Sostituisci *`MyClientDevice*`* con il nome dell'AWS IoToggetto da connettere come dispositivo client. Puoi anche specificare un nome con un carattere `*` jolly che corrisponda al nome del dispositivo client. Il `*` carattere jolly deve essere alla fine del nome.

  Se devi connettere un secondo dispositivo client, *`MyOtherClientDevicesostituisci*`* con il nome di quel dispositivo client o con un pattern di caratteri jolly che corrisponda al nome del dispositivo client. Altrimenti, puoi rimuovere o mantenere questa sezione della regola di selezione che consente ai dispositivi client con nomi corrispondenti di `MyOtherClientDevice*` connettersi e comunicare.

- Questa politica utilizza un OR operatore per consentire anche ai dispositivi client i cui nomi iniziano con di `MyOtherClientDevice` connettersi e comunicare su tutti gli argomenti MQTT. È possibile rimuovere questa clausola nella regola di selezione o modificarla in modo che corrisponda ai dispositivi client da connettere.
- Questa politica consente ai dispositivi client di pubblicare e sottoscrivere tutti gli argomenti MQTT. Per seguire le migliori pratiche di sicurezza, `mqtt:subscribe` limitate le operazioni `mqtt:publish` e al set minimo di argomenti utilizzati dai dispositivi client per comunicare.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice* OR
 thingName: MyOtherClientDevice*",
        "policyName": "MyClientDevicePolicy"
      }
    },
    "policies": {
      "MyClientDevicePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish to all
 topics.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowSubscribe": {
```

```
          "statementDescription": "Allow client devices to subscribe to all
 topics.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

Per ulteriori informazioni, consulta Configurazione del componente di autenticazione del dispositivo client.

ii. Scegli Conferma.

d. Per il aws.greengrass.clientdevices.mqtt.Bridgecomponente, scegli Modifica configurazione.

e. Nella modalità Modifica configurazione per il componente bridge MQTT, configurate una mappatura degli argomenti che inoltri i messaggi MQTT dai dispositivi client a. AWS IoT Core Esegui questa operazione:

i. In Configurazione, nel blocco Configurazione per unire codice, immettete la seguente configurazione. Questa configurazione specifica di inoltrare i messaggi MQTT sul filtro degli `clients/+/hello/world` argomenti dai dispositivi client al servizio cloud. AWS IoT Core Ad esempio, questo filtro per argomenti corrisponde all'argomento. `clients/MyClientDevice1/hello/world`

```
{
  "mqttTopicMapping": {
    "HelloWorldIotCoreMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

Per ulteriori informazioni, vedere Configurazione dei componenti del bridge MQTT.

ii. Scegli Conferma.

8. Scegliete Review and deploy per esaminare la distribuzione creata da questa pagina per voi.
9. Se non hai precedentemente impostato il ruolo di servizio Greengrass in questa regione, la console apre una modalità per configurare il ruolo di servizio per te. Il componente di autenticazione dei dispositivi client utilizza questo ruolo di servizio per verificare l'identità dei dispositivi client e il componente del rilevatore IP utilizza questo ruolo di servizio per gestire le informazioni di base sulla connettività dei dispositivi. Scegli Concedi autorizzazioni.
10. Nella pagina di revisione, scegli Distribuisci per avviare la distribuzione sul dispositivo principale.
11. Per verificare che la distribuzione abbia esito positivo, controlla lo stato della distribuzione e controlla i log sul dispositivo principale. Per verificare lo stato della distribuzione sul dispositivo principale, puoi scegliere Target nella panoramica della distribuzione. Per ulteriori informazioni, consulta gli argomenti seguenti:

    - Controllo dello stato di implementazione
    - Monitora AWS IoT Greengrass i registri

Fase 3: Connect i dispositivi client

I dispositivi client possono utilizzare il SDK per dispositivi AWS IoT per rilevare, connettersi e comunicare con un dispositivo principale. Il dispositivo client deve essere una AWS IoT cosa. Per ulteriori informazioni, consulta Create a thing object nella AWS IoT CoreDeveloper Guide.

In questa sezione, installate la SDK per dispositivi AWS IoTversione 2 per Python ed eseguite l'applicazione di esempio Greengrass discovery da. SDK per dispositivi AWS IoT

> **Note**
>
> SDK per dispositivi AWS IoTÈ disponibile anche in altri linguaggi di programmazione. Questo tutorial utilizza la SDK per dispositivi AWS IoT versione 2 per Python, ma puoi esplorare gli altri SDK per il tuo caso d'uso. Per ulteriori informazioni, consulta AWS IoTDevice SDK nella Device Guide. AWS IoT Core

Per connettere un dispositivo client a un dispositivo principale

1. Scarica e installa la SDK per dispositivi AWS IoTversione 2 per Python AWS IoT sull'oggetto da connettere come dispositivo client.

Sul dispositivo client, procedi come segue:

a. Clona il repository SDK per dispositivi AWS IoT v2 for Python per scaricarlo.

```
git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

b. Installa la SDK per dispositivi AWS IoT v2 per Python.

```
python3 -m pip install --user ./aws-iot-device-sdk-python-v2
```

2. Passa alla cartella samples nella SDK per dispositivi AWS IoT v2 per Python.

```
cd aws-iot-device-sdk-python-v2/samples
```

3. Esegui l'applicazione Greengrass discovery di esempio. Questa applicazione prevede argomenti che specifichino il nome dell'oggetto del dispositivo client, l'argomento e il messaggio MQTT da utilizzare e i certificati che autenticano e proteggono la connessione. L'esempio seguente invia un messaggio Hello World all'argomento. `clients/`*`MyClientDevice1`*`/hello/world`

> **Note**
>
> Questo argomento corrisponde all'argomento in cui è stato configurato il bridge MQTT per inoltrare i messaggi in AWS IoT Core precedenza.

- Sostituisci *`MyClientDevice1`* con il nome dell'oggetto del dispositivo client.
- Sostituisci *`~/certs/ AmazonRoot CA1.pem con`* il percorso del certificato CA root Amazon sul dispositivo client.
- Sostituisci *`~/certs/device.pem.crt con il percorso del certificato del dispositivo`* sul dispositivo client.
- Sostituisci *`~/certs/private.pem.key`* con il percorso del file della chiave privata sul dispositivo client.
- Sostituisci *`us-east-1`* con la regione in cui operano il dispositivo client e il dispositivo principale. AWS

```
python3 basic_discovery.py \\
  --thing_name MyClientDevice1 \\
```

```
  --topic 'clients/MyClientDevice1/hello/world' \\
  --message 'Hello World!' \\
  --ca_file ~/certs/AmazonRootCA1.pem \\
  --cert ~/certs/device.pem.crt \\
  --key ~/certs/private.pem.key \\
  --region us-east-1 \\
  --verbosity Warn
```

L'applicazione di esempio Discovery invia il messaggio 10 volte e si disconnette. Inoltre, sottoscrive lo stesso argomento in cui pubblica i messaggi. Se l'output indica che l'applicazione ha ricevuto messaggi MQTT sull'argomento, il dispositivo client può comunicare correttamente con il dispositivo principale.

```
Performing greengrass discovery...
awsiot.greengrass_discovery.DiscoverResponse(gg_groups=[awsiot.greengrass_discovery.GGGroup
coreDevice-MyGreengrassCore',
 cores=[awsiot.greengrass_discovery.GGCore(thing_arn='arn:aws:iot:us-
east-1:123456789012:thing/MyGreengrassCore',
 connectivity=[awsiot.greengrass_discovery.ConnectivityInfo(id='203.0.113.0',
 host_address='203.0.113.0', metadata='', port=8883)])],
 certificate_authorities=['-----BEGIN CERTIFICATE-----\
MIICiT...EXAMPLE=\
-----END CERTIFICATE-----\
'])])
Trying core arn:aws:iot:us-east-1:123456789012:thing/MyGreengrassCore at host
 203.0.113.0 port 8883
Connected!
Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 0}

Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 0}'
Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 1}

Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 1}'

...

Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 9}
```

```
Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 9}'
```

Se invece l'applicazione genera un errore, consulta Risoluzione dei problemi di rilevamento di Greengrass.

Puoi anche visualizzare i log di Greengrass sul dispositivo principale per verificare se il dispositivo client si connette e invia messaggi correttamente. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

4. Verificate che il bridge MQTT inoltri i messaggi dal dispositivo client a. AWS IoT Core È possibile utilizzare il client di test MQTT nella AWS IoT Core console per sottoscrivere un filtro per argomenti MQTT. Esegui questa operazione:

   a. Passare alla console AWS IoT.

   b. Nel menu di navigazione a sinistra, sotto Test, scegli MQTT test client.

   c. Nella scheda Sottoscrivi a un argomento, per Filtro argomento, inserisci `clients/+/hello/world` per iscriverti ai messaggi del dispositivo client dal dispositivo principale.

   d. Scegliere Subscribe (Effettua sottoscrizione).

   e. Esegui nuovamente l'applicazione di pubblicazione/sottoscrizione sul dispositivo client.

      Il client di test MQTT visualizza i messaggi inviati dal dispositivo client su argomenti che corrispondono a questo filtro per argomenti.

Fase 4: Sviluppare un componente che comunichi con i dispositivi client

È possibile sviluppare componenti Greengrass che comunicano con i dispositivi client. I componenti utilizzano la comunicazione tra processi (IPC) e l'interfaccia locale di pubblicazione/sottoscrizione per comunicare su un dispositivo principale. Per interagire con i dispositivi client, configurate il componente bridge MQTT per inoltrare i messaggi tra i dispositivi client e l'interfaccia di pubblicazione/sottoscrizione locale.

In questa sezione, si aggiorna il componente bridge MQTT per inoltrare i messaggi dai dispositivi client all'interfaccia di pubblicazione/sottoscrizione locale. Quindi, sviluppate un componente che sottoscrive questi messaggi e stampa i messaggi quando li riceve.

Sviluppare un componente che comunichi con i dispositivi client

1. Rivedi la distribuzione sul dispositivo principale e configura il componente bridge MQTT per inoltrare i messaggi dai dispositivi client alla pubblicazione/sottoscrizione locale. Esegui questa operazione:

   a. Passare alla console AWS IoT Greengrass.

   b. Nel menu di navigazione a sinistra, scegli Dispositivi principali.

   c. Nella pagina Dispositivi principali, scegli il dispositivo principale che stai utilizzando per questo tutorial.

   d. Nella pagina dei dettagli del dispositivo principale, scegli la scheda Dispositivi client.

   e. Nella scheda Dispositivi client, scegli Configura cloud discovery.

      Viene visualizzata la pagina Configura il rilevamento dei dispositivi principali. In questa pagina è possibile modificare o configurare i componenti del dispositivo client da distribuire sul dispositivo principale.

   f. Nel passaggio 3, per il aws.greengrass.clientdevices.mqtt.Bridgecomponente, scegli Modifica configurazione.

   g. Nella modalità Modifica configurazione per il componente bridge MQTT, configurate una mappatura degli argomenti che inoltri i messaggi MQTT dai dispositivi client all'interfaccia di pubblicazione/sottoscrizione locale. Esegui questa operazione:

      i. In Configurazione, nella sezione Configurazione per unire il blocco di codice, immettete la seguente configurazione. Questa configurazione specifica di inoltrare messaggi MQTT su argomenti che corrispondono al filtro degli `clients/+/hello/world` argomenti dai dispositivi client al servizio AWS IoT Core cloud e al broker di pubblicazione/sottoscrizione Greengrass locale.

```
{
  "mqttTopicMapping": {
    "HelloWorldIotCoreMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "HelloWorldPubsubMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
```

```
      "target": "Pubsub"
    }
  }
}
```

Per ulteriori informazioni, vedere Configurazione dei componenti del bridge MQTT.

ii. Scegli Conferma.

h. Scegliete Review and deploy per esaminare la distribuzione creata da questa pagina per voi.

i. Nella pagina Revisione, scegli Implementa per avviare la distribuzione sul dispositivo principale.

j. Per verificare che la distribuzione abbia esito positivo, controlla lo stato della distribuzione e controlla i log sul dispositivo principale. Per verificare lo stato della distribuzione sul dispositivo principale, puoi scegliere Target nella panoramica della distribuzione. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Controllo dello stato di implementazione
- Monitora AWS IoT Greengrass i registri

2. Sviluppa e distribuisci un componente Greengrass che sottoscrive i messaggi Hello World dai dispositivi client. Esegui questa operazione:

a. Crea cartelle per ricette e artefatti sul dispositivo principale.

Linux or Unix

```
mkdir recipes
mkdir -p artifacts/com.example.clientdevices.MyHelloWorldSubscriber/1.0.0
```

Windows Command Prompt (CMD)

```
mkdir recipes
mkdir artifacts\com.example.clientdevices.MyHelloWorldSubscriber\1.0.0
```

PowerShell

```
mkdir recipes
mkdir artifacts\com.example.clientdevices.MyHelloWorldSubscriber\1.0.0
```

> **Important**
>
> È necessario utilizzare il seguente formato per il percorso della cartella degli artefatti. Includete il nome e la versione del componente specificati nella ricetta.
>
> ```
> artifacts/componentName/componentVersion/
> ```

b. Utilizzate un editor di testo per creare una ricetta di componenti con i seguenti contenuti. Questa ricetta specifica di installare la SDK per dispositivi AWS IoT v2 per Python ed eseguire uno script che sottoscrive l'argomento e stampa i messaggi.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

   ```
   nano recipes/com.example.clientdevices.MyHelloWorldSubscriber-1.0.0.json
   ```

   Copiate la seguente ricetta nel file.

   ```
   {
     "RecipeFormatVersion": "2020-01-25",
     "ComponentName": "com.example.clientdevices.MyHelloWorldSubscriber",
     "ComponentVersion": "1.0.0",
     "ComponentDescription": "A component that subscribes to Hello World messages
    from client devices.",
     "ComponentPublisher": "Amazon",
     "ComponentConfiguration": {
       "DefaultConfiguration": {
         "accessControl": {
           "aws.greengrass.ipc.pubsub": {
             "com.example.clientdevices.MyHelloWorldSubscriber:pubsub:1": {
               "policyDescription": "Allows access to subscribe to all topics.",
               "operations": [
                 "aws.greengrass#SubscribeToTopic"
               ],
               "resources": [
                 "*"
               ]
             }
           }
   ```

```
      }
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk",
        "run": "python3 -u {artifacts:path}/hello_world_subscriber.py"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk",
        "run": "py -3 -u {artifacts:path}/hello_world_subscriber.py"
      }
    }
  ]
}
```

c. Usa un editor di testo per creare un artefatto di script Python denominato `hello_world_subscriber.py` con i seguenti contenuti. Questa applicazione utilizza il servizio IPC publish/subscribe per sottoscrivere l'`clients/+/hello/world`argomento e stampare i messaggi che riceve.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

   ```
   nano artifacts/com.example.clientdevices.MyHelloWorldSubscriber/1.0.0/
   hello_world_subscriber.py
   ```

   Copia il seguente codice Python nel file.

   ```
   import sys
   import time
   import traceback
   ```

```
from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2

CLIENT_DEVICE_HELLO_WORLD_TOPIC = 'clients/+/hello/world'
TIMEOUT = 10


def on_hello_world_message(event):
    try:
        message = str(event.binary_message.message, 'utf-8')
        print('Received new message: %s' % message)
    except:
        traceback.print_exc()


try:
    ipc_client = GreengrassCoreIPCClientV2()

    # SubscribeToTopic returns a tuple with the response and the operation.
    _, operation = ipc_client.subscribe_to_topic(
        topic=CLIENT_DEVICE_HELLO_WORLD_TOPIC,
 on_stream_event=on_hello_world_message)
    print('Successfully subscribed to topic: %s' %
          CLIENT_DEVICE_HELLO_WORLD_TOPIC)

    # Keep the main thread alive, or the process will exit.
    try:
        while True:
            time.sleep(10)
    except InterruptedError:
        print('Subscribe interrupted.')

    operation.close()
except Exception:
    print('Exception occurred when using IPC.', file=sys.stderr)
    traceback.print_exc()
    exit(1)
```

> **Note**
>
> Questo componente utilizza il client IPC V2 nella SDK per dispositivi AWS IoTv2 for Python per comunicare con il software Core. AWS IoT Greengrass Rispetto al client

IPC originale, il client IPC V2 riduce la quantità di codice da scrivere per utilizzare IPC nei componenti personalizzati.

d. Usa la Greengrass CLI per distribuire il componente.

Linux or Unix

```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.clientdevices.MyHelloWorldSubscriber=1.0.0"
```

Windows Command Prompt (CMD)

```
C:\greengrass\v2/bin/greengrass-cli deployment create ^
  --recipeDir recipes ^
  --artifactDir artifacts ^
  --merge "com.example.clientdevices.MyHelloWorldSubscriber=1.0.0"
```

PowerShell

```
C:\greengrass\v2/bin/greengrass-cli deployment create `
  --recipeDir recipes `
  --artifactDir artifacts `
  --merge "com.example.clientdevices.MyHelloWorldSubscriber=1.0.0"
```

3. Visualizza i log dei componenti per verificare che il componente sia stato installato correttamente e che sottoscriva l'argomento.

Linux or Unix

```
sudo tail -f /greengrass/v2/logs/
com.example.clientdevices.MyHelloWorldSubscriber.log
```

PowerShell

```
gc C:\greengrass\v2/logs/com.example.clientdevices.MyHelloWorldSubscriber.log -
Tail 10 -Wait
```

È possibile mantenere aperto il feed di registro per verificare che il dispositivo principale riceva messaggi.

4. Sul dispositivo client, esegui nuovamente l'applicazione di rilevamento Greengrass di esempio per inviare messaggi al dispositivo principale.

```
python3 basic_discovery.py \\
  --thing_name MyClientDevice1 \\
  --topic 'clients/MyClientDevice1/hello/world' \\
  --message 'Hello World!' \\
  --ca_file ~/certs/AmazonRootCA1.pem \\
  --cert ~/certs/device.pem.crt \\
  --key ~/certs/private.pem.key \\
  --region us-east-1 \\
  --verbosity Warn
```

5. Visualizzate nuovamente i registri dei componenti per verificare che il componente riceva e stampi i messaggi dal dispositivo client.

   Linux or Unix

   ```
   sudo tail -f /greengrass/v2/logs/
   com.example.clientdevices.MyHelloWorldSubscriber.log
   ```

   PowerShell

   ```
   gc C:\greengrass\v2/logs/com.example.clientdevices.MyHelloWorldSubscriber.log -
   Tail 10 -Wait
   ```

Fase 5: Sviluppa un componente che interagisca con le ombre del dispositivo client

È possibile sviluppare componenti Greengrass che interagiscono con le ombre del AWS IoTdispositivo client. Un'ombra è un documento JSON che memorizza le informazioni sullo stato corrente o desiderato per qualsiasi AWS IoT cosa, ad esempio un dispositivo client. I componenti personalizzati possono accedere alle ombre dei dispositivi client per gestirne lo stato, anche quando il dispositivo client non è connesso. AWS IoT Ogni AWS IoT oggetto ha un'ombra senza nome e puoi anche creare più ombre con nome per ogni cosa.

In questa sezione, si distribuisce il componente Shadow Manager per gestire le ombre sul dispositivo principale. È inoltre necessario aggiornare il componente bridge MQTT per inoltrare messaggi shadow tra i dispositivi client e il componente shadow manager. Quindi, sviluppate un componente che aggiorna le ombre dei dispositivi client ed eseguite un'applicazione di esempio sui dispositivi client che risponde agli aggiornamenti shadow del componente. Questo componente rappresenta un'applicazione intelligente per la gestione della luce, in cui il dispositivo principale gestisce lo stato cromatico delle luci intelligenti che si collegano ad esso come dispositivi client.

Sviluppare un componente che interagisca con le ombre dei dispositivi client

1. Rivedi la distribuzione sul dispositivo principale per implementare il componente shadow manager e configura il componente bridge MQTT per inoltrare i messaggi shadow tra i dispositivi client e la pubblicazione/sottoscrizione locali, dove lo shadow manager comunica. Esegui questa operazione:

   a. Passare alla console AWS IoT Greengrass.

   b. Nel menu di navigazione a sinistra, scegli Dispositivi principali.

   c. Nella pagina Dispositivi principali, scegli il dispositivo principale che stai utilizzando per questo tutorial.

   d. Nella pagina dei dettagli del dispositivo principale, scegli la scheda Dispositivi client.

   e. Nella scheda Dispositivi client, scegli Configura cloud discovery.

      Viene visualizzata la pagina Configura il rilevamento dei dispositivi principali. In questa pagina è possibile modificare o configurare i componenti del dispositivo client da distribuire sul dispositivo principale.

   f. Nel passaggio 3, per il aws.greengrass.clientdevices.mqtt.Bridgecomponente, scegli Modifica configurazione.

   g. Nella modalità Modifica configurazione per il componente bridge MQTT, configurate una mappatura degli argomenti che inoltri i messaggi MQTT sugli argomenti shadow del dispositivo tra i dispositivi client e l'interfaccia di pubblicazione/sottoscrizione locale. Confermate inoltre che la distribuzione specifichi una versione del bridge MQTT compatibile. Il supporto shadow dei dispositivi client richiede MQTT bridge v2.2.0 o versione successiva. Esegui questa operazione:

      i. Per la versione Component, scegliete la versione 2.2.0 o successiva.

ii. In Configurazione, nel blocco Configurazione per unire codice, inserisci la seguente configurazione. Questa configurazione specifica di inoltrare messaggi MQTT su argomenti shadow.

```
{
  "mqttTopicMapping": {
    "HelloWorldIotCoreMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "HelloWorldPubsubMapping": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/#",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
}
```

Per ulteriori informazioni, vedere Configurazione dei componenti del bridge MQTT.

iii. Scegli Conferma.

h. Nel passaggio 3, selezionate il aws.greengrass.ShadowManagercomponente per distribuirlo.

i. Scegliete Review and deploy per esaminare la distribuzione creata da questa pagina per voi.

j. Nella pagina Revisione, scegli Implementa per avviare la distribuzione sul dispositivo principale.

k. Per verificare che la distribuzione abbia esito positivo, controlla lo stato della distribuzione e controlla i log sul dispositivo principale. Per verificare lo stato della distribuzione sul

dispositivo principale, puoi scegliere Target nella panoramica della distribuzione. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Controllo dello stato di implementazione
- Monitora AWS IoT Greengrass i registri

2. Sviluppa e implementa un componente Greengrass che gestisce i dispositivi client smart light. Esegui questa operazione:

   a. Crea una cartella con gli artefatti del componente sul dispositivo principale.

   Linux or Unix

   ```
   mkdir -p artifacts/com.example.clientdevices.MySmartLightManager/1.0.0
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir artifacts\com.example.clientdevices.MySmartLightManager\1.0.0
   ```

   PowerShell

   ```
   mkdir artifacts\com.example.clientdevices.MySmartLightManager\1.0.0
   ```

   > **Important**
   >
   > È necessario utilizzare il seguente formato per il percorso della cartella degli artefatti. Includete il nome e la versione del componente specificati nella ricetta.
   >
   > ```
   > artifacts/componentName/componentVersion/
   > ```

   b. Utilizzate un editor di testo per creare una ricetta di componenti con i seguenti contenuti. Questa ricetta specifica di installare la SDK per dispositivi AWS IoT v2 per Python ed eseguire uno script che interagisce con le ombre dei dispositivi client smart light per gestirne i colori.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il seguente comando per utilizzare GNU nano per creare il file.

```
nano recipes/com.example.clientdevices.MySmartLightManager-1.0.0.json
```

Copiate la seguente ricetta nel file.

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.clientdevices.MySmartLightManager",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that interacts with smart light client
 devices.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.Nucleus": {
      "VersionRequirement": "^2.6.0"
    },
    "aws.greengrass.ShadowManager": {
      "VersionRequirement": "^2.2.0"
    },
    "aws.greengrass.clientdevices.mqtt.Bridge": {
      "VersionRequirement": "^2.2.0"
    }
  },
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "smartLightDeviceNames": [],
      "accessControl": {
        "aws.greengrass.ShadowManager": {
          "com.example.clientdevices.MySmartLightManager:shadow:1": {
            "policyDescription": "Allows access to client devices' unnamed
 shadows",
            "operations": [
              "aws.greengrass#GetThingShadow",
              "aws.greengrass#UpdateThingShadow"
            ],
            "resources": [
              "$aws/things/MyClientDevice*/shadow"
            ]
          }
        },
        "aws.greengrass.ipc.pubsub": {
          "com.example.clientdevices.MySmartLightManager:pubsub:1": {
```

```
            "policyDescription": "Allows access to client devices' unnamed
 shadow updates",
            "operations": [
              "aws.greengrass#SubscribeToTopic"
            ],
            "resources": [
              "$aws/things/+/shadow/update/accepted"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk",
        "run": "python3 -u {artifacts:path}/smart_light_manager.py"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk",
        "run": "py -3 -u {artifacts:path}/smart_light_manager.py"
      }
    }
  ]
}
```

c. Usa un editor di testo per creare un artefatto di script Python denominato `smart_light_manager.py` con i seguenti contenuti. Questa applicazione utilizza il servizio IPC shadow per ottenere e aggiornare le ombre dei dispositivi client e il servizio IPC di pubblicazione/sottoscrizione locale per ricevere gli aggiornamenti shadow segnalati.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

```
nano artifacts/com.example.clientdevices.MySmartLightManager/1.0.0/
smart_light_manager.py
```

Copia il seguente codice Python nel file.

```
import json
import random
import sys
import time
import traceback
from uuid import uuid4

from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2
from awsiot.greengrasscoreipc.model import ResourceNotFoundError

SHADOW_COLOR_PROPERTY = 'color'
CONFIGURATION_CLIENT_DEVICE_NAMES = 'smartLightDeviceNames'
COLORS = ['red', 'orange', 'yellow', 'green', 'blue', 'purple']
SHADOW_UPDATE_TOPIC = '$aws/things/+/shadow/update/accepted'
SET_COLOR_INTERVAL = 15


class SmartLightDevice():
    def __init__(self, client_device_name: str, reported_color: str = None):
        self.name = client_device_name
        self.reported_color = reported_color
        self.desired_color = None


class SmartLightDeviceManager():
    def __init__(self, ipc_client: GreengrassCoreIPCClientV2):
        self.ipc_client = ipc_client
        self.devices = {}
        self.client_tokens = set()
        self.shadow_update_accepted_subscription_operation = None
        self.client_device_names_configuration_subscription_operation = None
        self.update_smart_light_device_list()

    def update_smart_light_device_list(self):
        # Update the device list from the component configuration.
        response = self.ipc_client.get_configuration(
            key_path=[CONFIGURATION_CLIENT_DEVICE_NAMES])
```

```
        # Identify the difference between the configuration and the currently
tracked devices.
        current_device_names = self.devices.keys()
        updated_device_names =
response.value[CONFIGURATION_CLIENT_DEVICE_NAMES]
        added_device_names = set(updated_device_names) -
set(current_device_names)
        removed_device_names = set(current_device_names) -
set(updated_device_names)
        # Stop tracking any smart light devices that are no longer in the
configuration.
        for name in removed_device_names:
            print('Removing %s from smart light device manager' % name)
            self.devices.pop(name)
        # Start tracking any new smart light devices that are in the
configuration.
        for name in added_device_names:
            print('Adding %s to smart light device manager' % name)
            device = SmartLightDevice(name)
            device.reported_color = self.get_device_reported_color(device)
            self.devices[name] = device
            print('Current color for %s is %s' % (name,
device.reported_color))

    def get_device_reported_color(self, smart_light_device):
        try:
            response = self.ipc_client.get_thing_shadow(
                thing_name=smart_light_device.name, shadow_name='')
            shadow = json.loads(str(response.payload, 'utf-8'))
            if 'reported' in shadow['state']:
                return shadow['state']['reported'].get(SHADOW_COLOR_PROPERTY)
            return None
        except ResourceNotFoundError:
            return None

    def request_device_color_change(self, smart_light_device, color):
        # Generate and track a client token for the request.
        client_token = str(uuid4())
        self.client_tokens.add(client_token)
        # Create a shadow payload, which must be a blob.
        payload_json = {
            'state': {
                'desired': {
                    SHADOW_COLOR_PROPERTY: color
```

```
                }
            },
            'clientToken': client_token
        }
        payload = bytes(json.dumps(payload_json), 'utf-8')
        self.ipc_client.update_thing_shadow(
            thing_name=smart_light_device.name, shadow_name='',
payload=payload)
        smart_light_device.desired_color = color

    def subscribe_to_shadow_update_accepted_events(self):
        if self.shadow_update_accepted_subscription_operation == None:
            # SubscribeToTopic returns a tuple with the response and the
operation.
            _, self.shadow_update_accepted_subscription_operation =
self.ipc_client.subscribe_to_topic(
                topic=SHADOW_UPDATE_TOPIC,
on_stream_event=self.on_shadow_update_accepted_event)
            print('Successfully subscribed to shadow update accepted topic')

    def close_shadow_update_accepted_subscription(self):
        if self.shadow_update_accepted_subscription_operation is not None:
            self.shadow_update_accepted_subscription_operation.close()

    def on_shadow_update_accepted_event(self, event):
        try:
            message = str(event.binary_message.message, 'utf-8')
            accepted_payload = json.loads(message)
            # Check for reported states from smart light devices and ignore
desired states from components.
            if 'reported' in accepted_payload['state']:
                # Process this update only if it uses a client token created by
this component.
                client_token = accepted_payload.get('clientToken')
                if client_token is not None and client_token in
self.client_tokens:
                    self.client_tokens.remove(client_token)
                    shadow_state = accepted_payload['state']['reported']
                    if SHADOW_COLOR_PROPERTY in shadow_state:
                        reported_color = shadow_state[SHADOW_COLOR_PROPERTY]
                        topic = event.binary_message.context.topic
                        client_device_name = topic.split('/')[2]
                        if client_device_name in self.devices:
```

```
                            # Set the reported color for the smart light
device.
                            self.devices[client_device_name].reported_color =
reported_color
                            print(
                                'Received shadow update confirmation from
client device: %s' % client_device_name)
                    else:
                        print("Shadow update doesn't specify color")
        except:
            traceback.print_exc()

    def subscribe_to_client_device_name_configuration_updates(self):
        if self.client_device_names_configuration_subscription_operation ==
None:
            # SubscribeToConfigurationUpdate returns a tuple with the response
and the operation.
            _, self.client_device_names_configuration_subscription_operation =
self.ipc_client.subscribe_to_configuration_update(
                key_path=[CONFIGURATION_CLIENT_DEVICE_NAMES],
on_stream_event=self.on_client_device_names_configuration_update_event)
            print(
                'Successfully subscribed to configuration updates for smart
light device names')

    def close_client_device_names_configuration_subscription(self):
        if self.client_device_names_configuration_subscription_operation is not
None:

self.client_device_names_configuration_subscription_operation.close()

    def on_client_device_names_configuration_update_event(self, event):
        try:
            if CONFIGURATION_CLIENT_DEVICE_NAMES in
event.configuration_update_event.key_path:
                print('Received configuration update for list of client
devices')
                self.update_smart_light_device_list()
        except:
            traceback.print_exc()


def choose_random_color():
    return random.choice(COLORS)
```

```
def main():
    try:
        # Create an IPC client and a smart light device manager.
        ipc_client = GreengrassCoreIPCClientV2()
        smart_light_manager = SmartLightDeviceManager(ipc_client)
        smart_light_manager.subscribe_to_shadow_update_accepted_events()

 smart_light_manager.subscribe_to_client_device_name_configuration_updates()
        try:
            # Keep the main thread alive, or the process will exit.
            while True:
                # Set each smart light device to a random color at a regular
 interval.
                for device_name in smart_light_manager.devices:
                    device = smart_light_manager.devices[device_name]
                    desired_color = choose_random_color()
                    print('Chose random color (%s) for %s' %
                          (desired_color, device_name))
                    if desired_color == device.desired_color:
                        print('Desired color for %s is already %s' %
                              (device_name, desired_color))
                    elif desired_color == device.reported_color:
                        print('Reported color for %s is already %s' %
                              (device_name, desired_color))
                    else:
                        smart_light_manager.request_device_color_change(
                            device, desired_color)
                        print('Requested color change for %s to %s' %
                              (device_name, desired_color))
                time.sleep(SET_COLOR_INTERVAL)
        except InterruptedError:
            print('Application interrupted')
        smart_light_manager.close_shadow_update_accepted_subscription()

 smart_light_manager.close_client_device_names_configuration_subscription()
    except Exception:
        print('Exception occurred', file=sys.stderr)
        traceback.print_exc()
        exit(1)


if __name__ == '__main__':
```

```
    main()
```

Questa applicazione Python esegue le seguenti operazioni:

- Legge la configurazione del componente per ottenere l'elenco dei dispositivi client smart light da gestire.
- Sottoscrive le notifiche di aggiornamento della configurazione utilizzando l'operazione SubscribeToConfigurationUpdate IPC. Il software AWS IoT Greengrass Core invia notifiche ogni volta che la configurazione del componente cambia. Quando il componente riceve una notifica di aggiornamento della configurazione, aggiorna l'elenco dei dispositivi client smart light che gestisce.
- Ottiene l'ombra di ogni dispositivo client smart light per ottenere lo stato cromatico iniziale.
- Imposta il colore di ogni dispositivo client smart light su un colore casuale ogni 15 secondi. Il componente aggiorna l'ombra del dispositivo client per cambiarne il colore. Questa operazione invia un evento shadow delta al dispositivo client tramite MQTT.
- Effettua la sottoscrizione ai messaggi Shadow Update accettati sull'interfaccia di pubblicazione/sottoscrizione locale utilizzando l'operazione IPC. SubscribeToTopic Questo componente riceve questi messaggi per tracciare il colore di ogni dispositivo client smart light. Quando un dispositivo client smart light riceve un aggiornamento ombra, invia un messaggio MQTT per confermare che ha ricevuto l'aggiornamento. Il bridge MQTT inoltra questo messaggio all'interfaccia di pubblicazione/sottoscrizione locale.

d. Usa la Greengrass CLI per distribuire il componente. Quando si distribuisce questo componente, si specifica l'elenco dei dispositivi client di cui gestisce le `smartLightDeviceNames` ombre. Sostituisci *`MyClientDevice1`* con il nome dell'oggetto del dispositivo client.

   Linux or Unix

```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.clientdevices.MySmartLightManager=1.0.0" \
  --update-config '{
    "com.example.clientdevices.MySmartLightManager": {
      "MERGE": {
        "smartLightDeviceNames": [
          "MyClientDevice1"
        ]
```

```
      }
    }
  }'
```

Windows Command Prompt (CMD)

```
C:\greengrass\v2/bin/greengrass-cli deployment create ^
  --recipeDir recipes ^
  --artifactDir artifacts ^
  --merge "com.example.clientdevices.MySmartLightManager=1.0.0" ^
  --update-config '{"com.example.clientdevices.MySmartLightManager":
{"MERGE":{"smartLightDeviceNames":["MyClientDevice1"]}}}'
```

PowerShell

```
C:\greengrass\v2/bin/greengrass-cli deployment create `
  --recipeDir recipes `
  --artifactDir artifacts `
  --merge "com.example.clientdevices.MySmartLightManager=1.0.0" `
  --update-config '{
    "com.example.clientdevices.MySmartLightManager": {
      "MERGE": {
        "smartLightDeviceNames": [
          "MyClientDevice1"
        ]
      }
    }
  }'
```

3. Visualizza i log dei componenti per verificare che il componente sia installato ed eseguito correttamente.

   Linux or Unix

   ```
   sudo tail -f /greengrass/v2/logs/
   com.example.clientdevices.MySmartLightManager.log
   ```

   PowerShell

   ```
   gc C:\greengrass\v2/logs/com.example.clientdevices.MySmartLightManager.log -Tail
    10 -Wait
   ```

Il componente invia richieste per cambiare il colore del dispositivo client smart light. Il gestore delle ombre riceve la richiesta e imposta lo `desired` stato dell'ombra. Tuttavia, il dispositivo client smart light non è ancora in esecuzione, quindi lo `reported` stato dell'ombra non cambia. I log del componente includono i seguenti messaggi.

```
2022-07-07T03:49:24.908Z [INFO] (Copier)
 com.example.clientdevices.MySmartLightManager: stdout. Chose random color (blue)
 for MyClientDevice1.
 {scriptName=services.com.example.clientdevices.MySmartLightManager.lifecycle.Run,
 serviceName=com.example.clientdevices.MySmartLightManager, currentState=RUNNING}
2022-07-07T03:49:24.912Z [INFO] (Copier)
 com.example.clientdevices.MySmartLightManager: stdout.
 Requested color change for MyClientDevice1 to blue.
 {scriptName=services.com.example.clientdevices.MySmartLightManager.lifecycle.Run,
 serviceName=com.example.clientdevices.MySmartLightManager, currentState=RUNNING}
```

È possibile tenere aperto il feed di registro per vedere quando il componente stampa i messaggi.

4. Scarica ed esegui un'applicazione di esempio che utilizza Greengrass discovery e sottoscrive gli aggiornamenti shadow del dispositivo. Sul dispositivo client, effettuate le seguenti operazioni:

   a. Passa alla cartella samples nella SDK per dispositivi AWS IoT v2 per Python. Questa applicazione di esempio utilizza un modulo di analisi della riga di comando nella cartella samples.

      ```
      cd aws-iot-device-sdk-python-v2/samples
      ```

   b. Usa un editor di testo per creare uno script Python denominato `basic_discovery_shadow.py` con i seguenti contenuti. Questa applicazione utilizza Greengrass discovery e shadows per mantenere sincronizzata una proprietà tra il dispositivo client e il dispositivo principale.

      Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

      ```
      nano basic_discovery_shadow.py
      ```

      Copia il seguente codice Python nel file.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0.

from awscrt import io
from awscrt import mqtt
from awsiot import iotshadow
from awsiot.greengrass_discovery import DiscoveryClient
from awsiot import mqtt_connection_builder
from concurrent.futures import Future
import sys
import threading
import traceback
from uuid import uuid4

# Parse arguments
import utils.command_line_utils;
cmdUtils = utils.command_line_utils.CommandLineUtils("Basic Discovery -
 Greengrass discovery example with device shadows.")
cmdUtils.add_common_mqtt_commands()
cmdUtils.add_common_topic_message_commands()
cmdUtils.add_common_logging_commands()
cmdUtils.register_command("key", "<path>", "Path to your key in PEM format.",
 True, str)
cmdUtils.register_command("cert", "<path>", "Path to your client certificate in
 PEM format.", True, str)
cmdUtils.remove_command("endpoint")
cmdUtils.register_command("thing_name", "<str>", "The name assigned to your IoT
 Thing", required=True)
cmdUtils.register_command("region", "<str>", "The region to connect through.",
 required=True)
cmdUtils.register_command("shadow_property", "<str>", "The name of the shadow
 property you want to change (optional, default='color'", default="color")
# Needs to be called so the command utils parse the commands
cmdUtils.get_args()

# Using globals to simplify sample code
is_sample_done = threading.Event()
mqtt_connection = None
shadow_thing_name = cmdUtils.get_command_required("thing_name")
shadow_property = cmdUtils.get_command("shadow_property")

SHADOW_VALUE_DEFAULT = "off"
```

```
class LockedData:
    def __init__(self):
        self.lock = threading.Lock()
        self.shadow_value = None
        self.disconnect_called = False
        self.request_tokens = set()

locked_data = LockedData()


def on_connection_interupted(connection, error, **kwargs):
    print('connection interrupted with error {}'.format(error))


def on_connection_resumed(connection, return_code, session_present, **kwargs):
    print('connection resumed with return code {}, session present
 {}'.format(return_code, session_present))

# Try IoT endpoints until we find one that works
def try_iot_endpoints():
    for gg_group in discover_response.gg_groups:
        for gg_core in gg_group.cores:
            for connectivity_info in gg_core.connectivity:
                try:
                    print('Trying core {} at host {} port
 {}'.format(gg_core.thing_arn, connectivity_info.host_address,
 connectivity_info.port))
                    mqtt_connection = mqtt_connection_builder.mtls_from_path(
                        endpoint=connectivity_info.host_address,
                        port=connectivity_info.port,
                        cert_filepath=cmdUtils.get_command_required("cert"),
                        pri_key_filepath=cmdUtils.get_command_required("key"),

 ca_bytes=gg_group.certificate_authorities[0].encode('utf-8'),
                        on_connection_interrupted=on_connection_interupted,
                        on_connection_resumed=on_connection_resumed,
                        client_id=cmdUtils.get_command_required("thing_name"),
                        clean_session=False,
                        keep_alive_secs=30)

                    connect_future = mqtt_connection.connect()
                    connect_future.result()
                    print('Connected!')
                    return mqtt_connection
```

```
                except Exception as e:
                    print('Connection failed with exception {}'.format(e))
                    continue

    exit('All connection attempts failed')

# Function for gracefully quitting this sample
def exit(msg_or_exception):
    if isinstance(msg_or_exception, Exception):
        print("Exiting sample due to exception.")
        traceback.print_exception(msg_or_exception.__class__, msg_or_exception,
 sys.exc_info()[2])
    else:
        print("Exiting sample:", msg_or_exception)

    with locked_data.lock:
        if not locked_data.disconnect_called:
            print("Disconnecting...")
            locked_data.disconnect_called = True
            future = mqtt_connection.disconnect()
            future.add_done_callback(on_disconnected)

def on_disconnected(disconnect_future):
    # type: (Future) -> None
    print("Disconnected.")

    # Signal that sample is finished
    is_sample_done.set()

def on_get_shadow_accepted(response):
    # type: (iotshadow.GetShadowResponse) -> None
    try:
        with locked_data.lock:
            # check that this is a response to a request from this session
            try:
                locked_data.request_tokens.remove(response.client_token)
            except KeyError:
                return

            print("Finished getting initial shadow state.")
            if locked_data.shadow_value is not None:
                print("  Ignoring initial query because a delta event has
 already been received.")
```

```
                return

        if response.state:
            if response.state.delta:
                value = response.state.delta.get(shadow_property)
                if value:
                    print("  Shadow contains delta value '{}'.".format(value))
                    change_shadow_value(value)
                    return

            if response.state.reported:
                value = response.state.reported.get(shadow_property)
                if value:
                    print("  Shadow contains reported value
 '{}'.".format(value))

 set_local_value_due_to_initial_query(response.state.reported[shadow_property])
                    return

        print("  Shadow document lacks '{}' property. Setting
 defaults...".format(shadow_property))
        change_shadow_value(SHADOW_VALUE_DEFAULT)
        return

    except Exception as e:
        exit(e)

def on_get_shadow_rejected(error):
    # type: (iotshadow.ErrorResponse) -> None
    try:
        # check that this is a response to a request from this session
        with locked_data.lock:
            try:
                locked_data.request_tokens.remove(error.client_token)
            except KeyError:
                return

        if error.code == 404:
            print("Thing has no shadow document. Creating with defaults...")
            change_shadow_value(SHADOW_VALUE_DEFAULT)
        else:
            exit("Get request was rejected. code:{} message:'{}'".format(
                error.code, error.message))
```

```
    except Exception as e:
        exit(e)

def on_shadow_delta_updated(delta):
    # type: (iotshadow.ShadowDeltaUpdatedEvent) -> None
    try:
        print("Received shadow delta event.")
        if delta.state and (shadow_property in delta.state):
            value = delta.state[shadow_property]
            if value is None:
                print("  Delta reports that '{}' was deleted. Resetting
 defaults...".format(shadow_property))
                change_shadow_value(SHADOW_VALUE_DEFAULT)
                return
            else:
                print("  Delta reports that desired value is '{}'. Changing
 local value...".format(value))
                if (delta.client_token is not None):
                    print ("  ClientToken is: " + delta.client_token)
                change_shadow_value(value, delta.client_token)
        else:
            print("  Delta did not report a change in
 '{}'".format(shadow_property))

    except Exception as e:
        exit(e)

def on_publish_update_shadow(future):
    #type: (Future) -> None
    try:
        future.result()
        print("Update request published.")
    except Exception as e:
        print("Failed to publish update request.")
        exit(e)

def on_update_shadow_accepted(response):
    # type: (iotshadow.UpdateShadowResponse) -> None
    try:
        # check that this is a response to a request from this session
        with locked_data.lock:
            try:
                locked_data.request_tokens.remove(response.client_token)
            except KeyError:
```

```
                return

        try:
            if response.state.reported != None:
                if shadow_property in response.state.reported:
                    print("Finished updating reported shadow value to
 '{}'.".format(response.state.reported[shadow_property])) # type: ignore
                else:
                    print ("Could not find shadow property with name:
 '{}'.".format(shadow_property)) # type: ignore
            else:
                print("Shadow states cleared.") # when the shadow states are
 cleared, reported and desired are set to None
        except:
            exit("Updated shadow is missing the target property")

    except Exception as e:
        exit(e)

def on_update_shadow_rejected(error):
    # type: (iotshadow.ErrorResponse) -> None
    try:
        # check that this is a response to a request from this session
        with locked_data.lock:
            try:
                locked_data.request_tokens.remove(error.client_token)
            except KeyError:
                return

        exit("Update request was rejected. code:{} message:'{}'".format(
            error.code, error.message))

    except Exception as e:
        exit(e)

def set_local_value_due_to_initial_query(reported_value):
    with locked_data.lock:
        locked_data.shadow_value = reported_value

def change_shadow_value(value, token=None):
    with locked_data.lock:
        if locked_data.shadow_value == value:
            print("Local value is already '{}'.".format(value))
            return
```

```
        print("Changed local shadow value to '{}'.".format(value))
        locked_data.shadow_value = value

        print("Updating reported shadow value to '{}'...".format(value))

        reuse_token = token is not None
        # use a unique token so we can correlate this "request" message to
        # any "response" messages received on the /accepted and /rejected
topics
        if not reuse_token:
            token = str(uuid4())

        # if the value is "clear shadow" then send a UpdateShadowRequest with
None
        # for both reported and desired to clear the shadow document
completely.
        if value == "clear_shadow":
            tmp_state = iotshadow.ShadowState(reported=None, desired=None,
reported_is_nullable=True, desired_is_nullable=True)
            request = iotshadow.UpdateShadowRequest(
                thing_name=shadow_thing_name,
                state=tmp_state,
                client_token=token,
            )
        # Otherwise, send a normal update request
        else:
            # if the value is "none" then set it to a Python none object to
            # clear the individual shadow property
            if value == "none":
                value = None

            request = iotshadow.UpdateShadowRequest(
            thing_name=shadow_thing_name,
            state=iotshadow.ShadowState(
                reported={ shadow_property: value }
                ),
                client_token=token,
            )

        future = shadow_client.publish_update_shadow(request,
mqtt.QoS.AT_LEAST_ONCE)

        if not reuse_token:
```

```
            locked_data.request_tokens.add(token)

        future.add_done_callback(on_publish_update_shadow)


if __name__ == '__main__':
    tls_options =
 io.TlsContextOptions.create_client_with_mtls_from_path(cmdUtils.get_command_required("
 cmdUtils.get_command_required("key"))
    if cmdUtils.get_command(cmdUtils.m_cmd_ca_file):
        tls_options.override_default_trust_store_from_path(None,
 cmdUtils.get_command(cmdUtils.m_cmd_ca_file))
    tls_context = io.ClientTlsContext(tls_options)

    socket_options = io.SocketOptions()

    print('Performing greengrass discovery...')
    discovery_client =
 DiscoveryClient(io.ClientBootstrap.get_or_create_static_default(),
 socket_options, tls_context, cmdUtils.get_command_required("region"))
    resp_future =
 discovery_client.discover(cmdUtils.get_command_required("thing_name"))
    discover_response = resp_future.result()

    print(discover_response)
    if cmdUtils.get_command("print_discover_resp_only"):
        exit(0)

    mqtt_connection = try_iot_endpoints()
    shadow_client = iotshadow.IotShadowClient(mqtt_connection)

    try:
        # Subscribe to necessary topics.
        # Note that is **is** important to wait for "accepted/rejected"
 subscriptions
        # to succeed before publishing the corresponding "request".
        print("Subscribing to Update responses...")
        update_accepted_subscribed_future, _ =
 shadow_client.subscribe_to_update_shadow_accepted(

 request=iotshadow.UpdateShadowSubscriptionRequest(thing_name=shadow_thing_name),
            qos=mqtt.QoS.AT_LEAST_ONCE,
            callback=on_update_shadow_accepted)
```

```
        update_rejected_subscribed_future, _ =
shadow_client.subscribe_to_update_shadow_rejected(

request=iotshadow.UpdateShadowSubscriptionRequest(thing_name=shadow_thing_name),
            qos=mqtt.QoS.AT_LEAST_ONCE,
            callback=on_update_shadow_rejected)

        # Wait for subscriptions to succeed
        update_accepted_subscribed_future.result()
        update_rejected_subscribed_future.result()

        print("Subscribing to Get responses...")
        get_accepted_subscribed_future, _ =
shadow_client.subscribe_to_get_shadow_accepted(

request=iotshadow.GetShadowSubscriptionRequest(thing_name=shadow_thing_name),
            qos=mqtt.QoS.AT_LEAST_ONCE,
            callback=on_get_shadow_accepted)

        get_rejected_subscribed_future, _ =
shadow_client.subscribe_to_get_shadow_rejected(

request=iotshadow.GetShadowSubscriptionRequest(thing_name=shadow_thing_name),
            qos=mqtt.QoS.AT_LEAST_ONCE,
            callback=on_get_shadow_rejected)

        # Wait for subscriptions to succeed
        get_accepted_subscribed_future.result()
        get_rejected_subscribed_future.result()

        print("Subscribing to Delta events...")
        delta_subscribed_future, _ =
shadow_client.subscribe_to_shadow_delta_updated_events(

request=iotshadow.ShadowDeltaUpdatedSubscriptionRequest(thing_name=shadow_thing_name),
            qos=mqtt.QoS.AT_LEAST_ONCE,
            callback=on_shadow_delta_updated)

        # Wait for subscription to succeed
        delta_subscribed_future.result()

        # The rest of the sample runs asynchronously.

        # Issue request for shadow's current state.
```

```
        # The response will be received by the on_get_accepted() callback
        print("Requesting current shadow state...")

        with locked_data.lock:
            # use a unique token so we can correlate this "request" message to
            # any "response" messages received on the /accepted and /rejected
topics
            token = str(uuid4())

            publish_get_future = shadow_client.publish_get_shadow(

request=iotshadow.GetShadowRequest(thing_name=shadow_thing_name,
client_token=token),
                qos=mqtt.QoS.AT_LEAST_ONCE)

            locked_data.request_tokens.add(token)

        # Ensure that publish succeeds
        publish_get_future.result()

    except Exception as e:
        exit(e)

    # Wait for the sample to finish (user types 'quit', or an error occurs)
    is_sample_done.wait()
```

Questa applicazione Python esegue le seguenti operazioni:

- Utilizza Greengrass discovery per scoprire e connettersi al dispositivo principale.
- Richiede il documento shadow dal dispositivo principale per ottenere lo stato iniziale della proprietà.
- Si iscrive agli eventi shadow delta, che il dispositivo principale invia quando il `desired` valore della proprietà è diverso dal suo `reported` valore. Quando l'applicazione riceve un evento shadow delta, modifica il valore della proprietà e invia un aggiornamento al dispositivo principale per impostare il nuovo valore come valore`reported`.

Questa applicazione combina i campioni Greengrass discovery e shadow della SDK per dispositivi AWS IoT v2.

c. Esegui l'applicazione di esempio. Questa applicazione prevede argomenti che specifichino il nome dell'oggetto del dispositivo client, la proprietà shadow da utilizzare e i certificati che autenticano e proteggono la connessione.

- Sostituisci *MyClientDevice1* con il nome dell'oggetto del dispositivo client.
- Sostituisci *~/certs/ AmazonRoot CA1.pem con* il percorso del certificato CA root Amazon sul dispositivo client.
- Sostituisci *~/certs/device.pem.crt con il percorso del certificato del dispositivo* sul dispositivo client.
- Sostituisci *~/certs/private.pem.key* con il percorso del file della chiave privata sul dispositivo client.
- Sostituisci *us-east-1* con la regione in cui operano il dispositivo client e il dispositivo principale. AWS

```
python3 basic_discovery_shadow.py \
  --thing_name MyClientDevice1 \
  --shadow_property color \
  --ca_file ~/certs/AmazonRootCA1.pem \
  --cert ~/certs/device.pem.crt \
  --key ~/certs/private.pem.key \
  --region us-east-1 \
  --verbosity Warn
```

L'applicazione di esempio sottoscrive gli argomenti shadow e attende di ricevere gli eventi shadow delta dal dispositivo principale. Se l'output indica che l'applicazione riceve e risponde agli eventi shadow delta, il dispositivo client può interagire correttamente con la relativa ombra sul dispositivo principale.

```
Performing greengrass discovery...
awsiot.greengrass_discovery.DiscoverResponse(gg_groups=[awsiot.greengrass_discovery.GGG
coreDevice-MyGreengrassCore',
 cores=[awsiot.greengrass_discovery.GGCore(thing_arn='arn:aws:iot:us-
east-1:123456789012:thing/MyGreengrassCore',
 connectivity=[awsiot.greengrass_discovery.ConnectivityInfo(id='203.0.113.0',
 host_address='203.0.113.0', metadata='', port=8883)])],
 certificate_authorities=['-----BEGIN CERTIFICATE-----
\nMIICiT...EXAMPLE=\n-----END CERTIFICATE-----\n'])])
```

```
Trying core arn:aws:iot:us-east-1:123456789012:thing/MyGreengrassCore at host
 203.0.113.0 port 8883
Connected!
Subscribing to Update responses...
Subscribing to Get responses...
Subscribing to Delta events...
Requesting current shadow state...
Received shadow delta event.
  Delta reports that desired value is 'purple'. Changing local value...
  ClientToken is: 3dce4d3f-e336-41ac-aa4f-7882725f0033
Changed local shadow value to 'purple'.
Updating reported shadow value to 'purple'...
Update request published.
```

Se invece l'applicazione genera un errore, consulta Risoluzione dei problemi di rilevamento di Greengrass.

Puoi anche visualizzare i log di Greengrass sul dispositivo principale per verificare se il dispositivo client si connette e invia messaggi correttamente. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

5. Visualizza nuovamente i registri dei componenti per verificare che il componente riceva conferme di aggiornamento ombra dal dispositivo client smart light.

   Linux or Unix

   ```
   sudo tail -f /greengrass/v2/logs/
   com.example.clientdevices.MySmartLightManager.log
   ```

   PowerShell

   ```
   gc C:\greengrass\v2/logs/com.example.clientdevices.MySmartLightManager.log -Tail
    10 -Wait
   ```

   Il componente registra i messaggi per confermare che il dispositivo client smart light ha cambiato colore.

   ```
   2022-07-07T03:49:24.908Z [INFO] (Copier)
    com.example.clientdevices.MySmartLightManager: stdout. Chose random color (blue)
    for MyClientDevice1.
   ```

```
 {scriptName=services.com.example.clientdevices.MySmartLightManager.lifecycle.Run,
 serviceName=com.example.clientdevices.MySmartLightManager, currentState=RUNNING}
2022-07-07T03:49:24.912Z [INFO] (Copier)
 com.example.clientdevices.MySmartLightManager: stdout.
 Requested color change for MyClientDevice1 to blue.
 {scriptName=services.com.example.clientdevices.MySmartLightManager.lifecycle.Run,
 serviceName=com.example.clientdevices.MySmartLightManager, currentState=RUNNING}
2022-07-07T03:49:24.959Z [INFO] (Copier)
 com.example.clientdevices.MySmartLightManager: stdout. Received
 shadow update confirmation from client device: MyClientDevice1.
 {scriptName=services.com.example.clientdevices.MySmartLightManager.lifecycle.Run,
 serviceName=com.example.clientdevices.MySmartLightManager, currentState=RUNNING}
```

> **Note**
>
> L'ombra del dispositivo client è sincronizzata tra il dispositivo principale e il dispositivo client. Tuttavia, il dispositivo principale non sincronizza l'ombra del dispositivo client conAWS IoT Core. Ad esempio, potresti sincronizzare un'ombra con AWS IoT Core per visualizzare o modificare lo stato di tutti i dispositivi del tuo parco dispositivi. Per ulteriori informazioni su come configurare il componente Shadow Manager con cui sincronizzare le ombreAWS IoT Core, consultaSincronizza le ombre del dispositivo locale con AWS IoT Core.

Hai completato questo tutorial. Il dispositivo client si connette al dispositivo principale, invia messaggi MQTT ai componenti Greengrass AWS IoT Core e riceve aggiornamenti shadow dal dispositivo principale. Per ulteriori informazioni sugli argomenti trattati in questo tutorial, consulta quanto segue:

- Associa i dispositivi client
- Gestisci gli endpoint principali dei dispositivi
- Verifica le comunicazioni con i dispositivi client
- API RESTful per la scoperta di Greengrass
- Inoltra messaggi MQTT tra dispositivi client e AWS IoT Core
- Interagisci con i dispositivi client nei componenti
- Interagisci con le ombre dei dispositivi
- Interazione e sincronizzazione delle ombre dei dispositivi client

Tutorial: Inizia a usare SageMaker Edge Manager

> **Important**
>
> SageMaker Edge Manager verrà interrotto il 26 aprile 2024. Per ulteriori informazioni su come continuare a distribuire i modelli sui dispositivi edge, consulta SageMaker Edge Manager End of Life.

Amazon SageMaker Edge Manager è un agente software che funziona su dispositivi edge. SageMaker Edge Manager fornisce la gestione dei modelli per i dispositivi edge in modo da poter impacchettare e utilizzare i modelli SageMaker compilati da Amazon NEO direttamente sui dispositivi core Greengrass. Utilizzando SageMaker Edge Manager, puoi anche campionare i dati di input e output del modello dai tuoi dispositivi principali e inviarli a loro Cloud AWS per il monitoraggio e l'analisi. Per ulteriori informazioni sul funzionamento di SageMaker Edge Manager sui dispositivi core Greengrass, vedere. Usa Amazon SageMaker Edge Manager sui dispositivi core Greengrass

Questo tutorial mostra come iniziare a utilizzare SageMaker Edge Manager con componenti AWS di esempio forniti su un dispositivo principale esistente. Questi componenti di esempio utilizzano il componente SageMaker Edge Manager come dipendenza per distribuire l'agente Edge Manager ed eseguono l'inferenza utilizzando modelli preaddestrati compilati utilizzando Neo. SageMaker Per ulteriori informazioni sull'agente SageMaker Edge Manager, consulta SageMaker Edge Manager nella Amazon SageMaker Developer Guide.

Per configurare e utilizzare l'agente SageMaker Edge Manager su un dispositivo principale Greengrass esistente, AWS fornisce codice di esempio che è possibile utilizzare per creare i seguenti componenti di inferenza e modello di esempio.

- Classificazione delle immagini
  - `com.greengrass.SageMakerEdgeManager.ImageClassification`
  - `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`
- Rilevamento di oggetti
  - `com.greengrass.SageMakerEdgeManager.ObjectDetection`
  - `com.greengrass.SageMakerEdgeManager.ObjectDetection.Model`

Questo tutorial mostra come distribuire i componenti di esempio e l'agente SageMaker Edge Manager.

Argomenti

Prerequisiti

Per completare questo tutorial, è necessario soddisfare i seguenti prerequisiti:

- Un dispositivo core Greengrass in esecuzione su Amazon Linux 2, una piattaforma Linux basata su Debian (x86_64 o Armv8) o Windows (x86_64). Se non lo hai, consultare Tutorial: Nozioni di base su AWS IoT Greengrass V2.
- Python 3.6 o versione successiva, inclusa `pip` la tua versione di Python, installato sul tuo dispositivo principale.
- L'API OpenGL GLX runtime `libgl1-mesa-glx` () installata sul dispositivo principale.
- Un utente AWS Identity and Access Management (IAM) con autorizzazioni di amministratore.
- Un computer di sviluppo simile a Windows, Mac o UNIX abilitato a Internet che soddisfa i seguenti requisiti:
  - Python 3.6 o successivo installato.
  - AWS CLIinstallato e configurato con le credenziali utente dell'amministratore IAM. Per ulteriori informazioni, consulta Installazione AWS CLI e configurazione di. AWS CLI
- I seguenti bucket S3 creati nello Regione AWS stesso Account AWS dispositivo principale Greengrass:
  - Un bucket S3 per archiviare gli artefatti inclusi nell'inferenza del campione e nei componenti del modello. Questo tutorial utilizza *`DOC-EXAMPLE-BUCKET1`* per fare riferimento a questo bucket.
  - Un bucket S3 che associ alla tua flotta di dispositivi edge. SageMaker SageMaker Edge Manager richiede un bucket S3 per creare la flotta di dispositivi edge e per archiviare dati di esempio derivanti dall'esecuzione dell'inferenza sul dispositivo. Questo tutorial utilizza *`DOC-EXAMPLE-BUCKET2`* per fare riferimento a questo bucket.

  Per informazioni sulla creazione di bucket S3, consulta Guida introduttiva ad Amazon S3.

- Il ruolo del dispositivo Greengrass è configurato con quanto segue:
  - Una relazione di fiducia che consente `credentials.iot.amazonaws.com` e consente `sagemaker.amazonaws.com` di assumere il ruolo, come illustrato nel seguente esempio di policy IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.iot.amazonaws.com"
       },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

  - La politica gestita da AmazonSageMakerEdgeDeviceFleetPolicyIAM.
  - La AmazonSageMakerFullAccesspolicy gestita da IAM.
  - L'`s3:GetObject`azione per il bucket S3 che contiene gli artefatti dei componenti, come illustrato nel seguente esempio di policy IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
  {
    "Action": [
      "s3:GetObject"
    ],
    "Resource": [
      "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
    ],
    "Effect": "Allow"
```

```
  }
  ]
}
```

Configura il tuo dispositivo principale Greengrass in SageMaker Edge Manager

Le flotte di dispositivi SageMaker Edge in Edge Manager sono raccolte di dispositivi raggruppati logicamente. Per utilizzare SageMaker Edge Manager conAWS IoT Greengrass, è necessario creare una flotta di dispositivi edge che utilizzi lo stesso alias di AWS IoT ruolo del dispositivo principale Greengrass su cui viene distribuito SageMaker l'agente Edge Manager. Quindi, è necessario registrare il dispositivo principale come parte di quel parco dispositivi.

Argomenti

- Crea una flotta di dispositivi edge
- Registra il tuo dispositivo Greengrass core

Crea una flotta di dispositivi edge

Per creare una flotta di dispositivi edge (console)

1. Nella SageMaker console Amazon, scegli Edge Manager, quindi scegli flotte di dispositivi Edge.
2. Nella pagina Flotte di dispositivi, scegli Crea flotta di dispositivi.
3. In Proprietà del parco dispositivi, procedi come segue:
   - Per Nome parco dispositivi, inserisci un nome per il tuo parco dispositivi.
   - Per il ruolo IAM, inserisci l'Amazon Resource Name (ARN) dell'alias del AWS IoT ruolo che hai specificato durante la configurazione del dispositivo principale Greengrass.
   - Disattiva l'opzione Crea alias del ruolo IAM.
4. Seleziona Avanti.
5. In Configurazione di output, per l'URI del bucket S3, inserisci l'URI del bucket S3 che desideri associare al parco dispositivi.
6. Seleziona Invia.

Registra il tuo dispositivo Greengrass core

Per registrare il dispositivo Greengrass core come dispositivo edge (console)

1. Nella SageMaker console Amazon, scegli Edge Manager, quindi scegli Dispositivi Edge.
2. Nella pagina Dispositivi, scegli Registra dispositivi.
3. In Proprietà del dispositivo, in Nome parco dispositivi, inserisci il nome del parco dispositivi che hai creato, quindi scegli Avanti.
4. Seleziona Avanti.
5. In Origine dispositivo, in Nome dispositivo, inserisci il nome dell'AWS IoToggetto del tuo dispositivo principale Greengrass.
6. Seleziona Invia.

Create i componenti di esempio

Per aiutarti a iniziare a utilizzare il componente SageMaker Edge Manager, AWS fornisce uno script Python GitHub che crea i componenti di inferenza e modello di esempio e li carica al posto tuo. Cloud AWS Completa i seguenti passaggi su un computer di sviluppo.

Per creare i componenti di esempio

1. Scarica l'archivio degli esempi di AWS IoT Greengrass componenti sul tuo computer GitHub di sviluppo.
2. Vai alla `/machine-learning/sagemaker-edge-manager` cartella scaricata.

   ```
   cd download-directory/machine-learning/sagemaker-edge-manager
   ```
3. Eseguite il comando seguente per creare e caricare i componenti di esempio inCloud AWS.

   ```
   python3 create_components.py -r region -b DOC-EXAMPLE-BUCKET
   ```

   Sostituisci la *regione* con quella Regione AWS in cui hai creato il tuo dispositivo principale Greengrass e sostituisci *DOC-EXAMPLE-BUCKET1 con il nome del bucket S3* per memorizzare gli artefatti dei componenti.

> **Note**
>
> Per impostazione predefinita, lo script crea componenti di esempio sia per la classificazione delle immagini che per l'inferenza del rilevamento degli oggetti. Per creare componenti solo per un tipo specifico di inferenza, specificate l'`-i` *`ImageClassification | ObjectDetection`*argomento.

I componenti di inferenza e modello di esempio da utilizzare con SageMaker Edge Manager vengono ora creati in. Account AWS Per visualizzare i componenti di esempio nella AWS IoT Greengrassconsole, scegli Componenti, quindi in I miei componenti, cerca i seguenti componenti:

- `com.greengrass.SageMakerEdgeManager.ImageClassification`
- `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`
- `com.greengrass.SageMakerEdgeManager.ObjectDetection`
- `com.greengrass.SageMakerEdgeManager.ObjectDetection.Model`

Esegui un'inferenza di classificazione delle immagini di esempio

Per eseguire l'inferenza della classificazione delle immagini utilizzando i componenti di esempio AWS forniti e l'agente SageMaker Edge Manager, è necessario distribuire questi componenti sul dispositivo principale. L'implementazione di questi componenti scarica un modello Resnet-50 pre-addestrato SageMaker compilato da NEO e installa l'agente Edge Manager sul dispositivo. SageMaker L'agente SageMaker Edge Manager carica il modello e pubblica i risultati dell'inferenza sull'argomento. `gg/sageMakerEdgeManager/image-classification` Per visualizzare questi risultati di inferenza, utilizzate il client AWS IoT MQTT nella AWS IoT console per abbonarvi a questo argomento.

Argomenti

- Iscriviti all'argomento delle notifiche
- Distribuite i componenti di esempio
- Visualizza i risultati dell'inferenza

Iscriviti all'argomento delle notifiche

In questo passaggio, configurate il client AWS IoT MQTT nella AWS IoT console per guardare i messaggi MQTT pubblicati dal componente di inferenza di esempio. Per impostazione predefinita, il componente pubblica i risultati di inferenza sull'argomento. `gg/sageMakerEdgeManager/image-classification` Abbonati a questo argomento prima di distribuire il componente sul tuo dispositivo principale Greengrass per vedere i risultati dell'inferenza quando il componente viene eseguito per la prima volta.

Per sottoscrivere l'argomento delle notifiche predefinito

1. Nel menu di navigazione della AWS IoTconsole, scegliete Test, MQTT test client.
2. In Sottoscrivi a un argomento, nella casella Nome argomento, inserisci**`gg/sageMakerEdgeManager/image-classification`**.
3. Scegliere Subscribe (Effettua sottoscrizione).

Distribuite i componenti di esempio

In questo passaggio, configuri e distribuisci i seguenti componenti sul tuo dispositivo principale:

- `aws.greengrass.SageMakerEdgeManager`
- `com.greengrass.SageMakerEdgeManager.ImageClassification`
- `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`

Per distribuire i componenti (console)

1. Nel menu di navigazione della AWS IoT Greengrassconsole, scegli Distribuzioni, quindi scegli la distribuzione per il dispositivo di destinazione che desideri modificare.
2. Nella pagina di distribuzione, scegli Rivedi, quindi scegli Rivedi distribuzione.
3. Nella pagina Specificare la destinazione, scegli Avanti.
4. Nella pagina Seleziona componenti, procedi come segue:

    a. In I miei componenti, seleziona i seguenti componenti:

    - `com.greengrass.SageMakerEdgeManager.ImageClassification`
    - `com.greengrass.SageMakerEdgeManager.ImageClassification.Model`

b. In Componenti pubblici, disattiva l'interruttore Mostra solo i componenti selezionati, quindi seleziona il `aws.greengrass.SageMakerEdgeManager` componente.

c. Seleziona Avanti.

5. Nella pagina Configura componenti, selezionate il `aws.greengrass.SageMakerEdgeManager` componente ed effettuate le seguenti operazioni.

a. Scegli Configura componente.

b. In Aggiornamento della configurazione, in Configurazione da unire, inserisci la seguente configurazione.

```
{
    "DeviceFleetName": "device-fleet-name",
    "BucketName": "DOC-EXAMPLE-BUCKET"
}
```

Sostituiscilo *`device-fleet-name`*con il nome del parco dispositivi edge che hai creato e sostituisci *`DOC-EXAMPLE-BUCKET con il nome del bucket`* S3 associato al tuo parco dispositivi.

c. Seleziona Conferma e scegli Avanti.

6. Nella pagina Configura impostazioni avanzate, mantieni le impostazioni di configurazione predefinite e scegli Avanti.

7. Nella pagina di revisione, scegli Deploy

Per distribuire i componenti (AWS CLI)

1. Sul computer di sviluppo, create un `deployment.json` file per definire la configurazione di distribuzione per i componenti di SageMaker Edge Manager. Questo file dovrebbe essere simile all'esempio seguente.

```
{
  "targetArn":"targetArn",
  "components": {
    "aws.greengrass.SageMakerEdgeManager": {
      "componentVersion": "1.0.x",
      "configurationUpdate": {
        "merge": "{\"DeviceFleetName\":\"device-fleet-name\",\"BucketName\":\"DOC-
EXAMPLE-BUCKET2\"}"
```

```
      }
    },
    "com.greengrass.SageMakerEdgeManager.ImageClassification": {
      "componentVersion": "1.0.x",
      "configurationUpdate": {
      }
    },
    "com.greengrass.SageMakerEdgeManager.ImageClassification.Model": {
      "componentVersion": "1.0.x",
      "configurationUpdate": {
      }
    },
  }
}
```

- Nel campo `targetArn`, sostituisci *`targetArn`* con l'ARN (Amazon Resource Name) dell'oggetto o del gruppo di oggetti a cui destinare la distribuzione, nel seguente formato:
  - Oggetto: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
  - Gruppo di oggetti: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*
- Nel `merge` campo, *`device-fleet-name`*sostituiscilo con il nome del parco dispositivi edge che hai creato. Quindi, sostituisci *`DOC-EXAMPLE-BUCKET2 con il nome del bucket`* S3 associato al tuo parco dispositivi.
- Sostituisci le versioni dei componenti di ogni componente con l'ultima versione disponibile.

2. Eseguire comando seguente per distribuire i componenti sul dispositivo:

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

La distribuzione può richiedere alcuni minuti. Nel passaggio successivo, controlla il registro dei componenti per verificare che la distribuzione sia stata completata correttamente e per visualizzare i risultati dell'inferenza.

Visualizza i risultati dell'inferenza

Dopo aver distribuito i componenti, è possibile visualizzare i risultati dell'inferenza nel registro dei componenti sul dispositivo principale Greengrass e nel client AWS IoT MQTT nella console. AWS IoT

Per sottoscrivere l'argomento in cui il componente pubblica i risultati dell'inferenza, vedere. Iscriviti all'argomento delle notifiche

- AWS IoTClient MQTT: per visualizzare i risultati pubblicati dal componente di inferenza sull'argomento delle notifiche predefinite, completate i seguenti passaggi:

  1. Nel menu di navigazione della AWS IoTconsole, scegliete Test, MQTT test client.
  2. In Abbonamenti, scegli. **`gg/sageMakerEdgeManager/image-classification`**

- Registro dei componenti: per visualizzare i risultati dell'inferenza nel registro dei componenti, esegui il seguente comando sul tuo dispositivo principale Greengrass.

```
sudo tail -f /greengrass/v2/logs/
com.greengrass.SageMakerEdgeManager.ImageClassification.log
```

Se non riesci a visualizzare i risultati dell'inferenza nel registro dei componenti o nel client MQTT, la distribuzione non è riuscita o non ha raggiunto il dispositivo principale. Ciò può verificarsi se il dispositivo principale non è connesso a Internet o non dispone delle autorizzazioni necessarie per eseguire il componente. Esegui il comando seguente sul tuo dispositivo principale per visualizzare il file di registro del software AWS IoT Greengrass Core. Questo file include i log del servizio di distribuzione del dispositivo principale Greengrass.

```
sudo tail -f /greengrass/v2/logs/greengrass.log
```

Per ulteriori informazioni, consulta Risoluzione dei problemi di inferenza dell'apprendimento automatico.

Tutorial: eseguire l'inferenza della classificazione delle immagini di esempio utilizzando TensorFlow Lite

Questo tutorial mostra come utilizzare il componente di inferenza della classificazione delle immagini TensorFlow Lite per eseguire l'inferenza della classificazione delle immagini di esempio su un dispositivo principale Greengrass. Questo componente include le seguenti dipendenze dei componenti:

- TensorFlow Componente di archiviazione del modello di classificazione delle immagini Lite
- TensorFlow Componente di runtime Lite

Quando si distribuisce questo componente, scarica un modello MobileNet v1 pre-addestrato e installa il runtime TensorFlow Lite e le sue dipendenze. Questo componente pubblica i risultati dell'inferenza sull'argomento. `ml/tflite/image-classification` Per visualizzare questi risultati di inferenza, utilizzate il client AWS IoT MQTT nella AWS IoT console per abbonarvi a questo argomento.

In questo tutorial si implementa il componente di inferenza del campione per eseguire la classificazione delle immagini sull'immagine di esempio fornita da. AWS IoT Greengrass Dopo aver completato questo tutorial, puoi completareTutorial: Esegui l'inferenza della classificazione delle immagini di esempio sulle immagini di una fotocamera utilizzando TensorFlow Lite, che mostra come modificare il componente di inferenza del campione per eseguire la classificazione delle immagini sulle immagini di una fotocamera localmente su un dispositivo principale Greengrass.

Per ulteriori informazioni sull'apprendimento automatico sui dispositivi Greengrass, vedere. Esecuzione dell'inferenza di Machine Learning

Argomenti

Prerequisiti

Per completare questo tutorial, è necessario quanto segue:

- Un dispositivo core Linux Greengrass. Se non lo hai, consultare Tutorial: Nozioni di base su AWS IoT Greengrass V2. Il dispositivo principale deve soddisfare i seguenti requisiti:
  - Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
  - Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

```
sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
 libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

    Per abilitare lo stack di telecamere precedente

    1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleziona Opzioni di interfaccia.
    3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
    4. Riavvia il dispositivo Raspberry Pi.

Passaggio 1: iscriviti all'argomento delle notifiche predefinito

In questo passaggio, configurate il client AWS IoT MQTT nella AWS IoT console per guardare i messaggi MQTT pubblicati dal componente di classificazione delle immagini TensorFlow Lite. Per impostazione predefinita, il componente pubblica i risultati di inferenza sull'argomento. `ml/tflite/image-classification` Abbonati a questo argomento prima di distribuire il componente sul tuo dispositivo principale Greengrass per vedere i risultati dell'inferenza quando il componente viene eseguito per la prima volta.

Per sottoscrivere l'argomento delle notifiche predefinito

1. Nel menu di navigazione della AWS IoTconsole, scegliete Test, MQTT test client.

2. In Sottoscrivi a un argomento, nella casella Nome argomento, inserisci**`ml/tflite/image-classification`**.
3. Scegliere Subscribe (Effettua sottoscrizione).

Fase 2: Implementazione del componente di classificazione delle immagini TensorFlow Lite

In questo passaggio, distribuisci il componente di classificazione delle immagini TensorFlow Lite sul tuo dispositivo principale:

Per distribuire il componente di classificazione delle immagini TensorFlow Lite (console)

1. Nel menu di navigazione AWS IoT Greengrassdella console, scegli Componenti.
2. Nella pagina Componenti, nella scheda Componenti pubblici, scegli `aws.greengrass.TensorFlowLiteImageClassification`.
3. Nella pagina aws.greengrass.TensorFlowLiteImageClassification, scegli (Distribuisci).
4. Da Aggiungi alla distribuzione, scegli una delle seguenti opzioni:
   a. Per unire questo componente a una distribuzione esistente sul dispositivo di destinazione, scegli Aggiungi alla distribuzione esistente, quindi seleziona la distribuzione che desideri modificare.
   b. Per creare una nuova distribuzione sul dispositivo di destinazione, scegli Crea nuova distribuzione. Se hai una distribuzione esistente sul tuo dispositivo, la scelta di questo passaggio sostituisce la distribuzione esistente.
5. Nella pagina Specifica destinazione, procedi come segue:
   a. In Informazioni sulla distribuzione, inserisci o modifica il nome descrittivo della distribuzione.
   b. In Destinazione della distribuzione, seleziona una destinazione della distribuzione e scegli Avanti. Non è possibile modificare la destinazione della distribuzione se si sta revisionando una distribuzione esistente.
6. Nella pagina Seleziona componenti, in Componenti pubblici, verifica che il `aws.greengrass.TensorFlowLiteImageClassification` componente sia selezionato e scegli Avanti.
7. Nella pagina Configura componenti, mantieni le impostazioni di configurazione predefinite e scegli Avanti.

8. Nella pagina Configura impostazioni avanzate, mantieni le impostazioni di configurazione predefinite e scegli Avanti.
9. Nella pagina Revisione, scegli Distribuisci

Per distribuire il componente di classificazione delle immagini TensorFlow Lite () AWS CLI

1. Crea un `deployment.json` file per definire la configurazione di distribuzione per il componente di classificazione delle immagini TensorFlow Lite. Questo file dovrebbe avere il seguente aspetto:

```
{
  "targetArn":"targetArn",
  "components": {
    "aws.greengrass.TensorFlowLiteImageClassification": {
      "componentVersion": 2.1.0,
      "configurationUpdate": {
      }
    }
  }
}
```

   - Nel campo `targetArn`, sostituisci *`targetArn`* con l'ARN (Amazon Resource Name) dell'oggetto o del gruppo di oggetti a cui destinare la distribuzione, nel seguente formato:
     - Oggetto: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Gruppo di oggetti: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*
   - Questo tutorial utilizza la versione 2.1.0 del componente. Nell'oggetto `aws.greengrass.TensorFlowLiteObjectDetection` componente, sostituite *`2.1.0`* per utilizzare una versione diversa del componente TensorFlow Lite per il rilevamento degli oggetti.
2. Esegui il comando seguente per distribuire il componente di classificazione delle immagini TensorFlow Lite sul dispositivo:

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

La distribuzione può richiedere alcuni minuti. Nel passaggio successivo, controlla il registro dei componenti per verificare che la distribuzione sia stata completata correttamente e per visualizzare i risultati dell'inferenza.

Fase 3: Visualizzazione dei risultati dell'inferenza

Dopo aver distribuito il componente, è possibile visualizzare i risultati dell'inferenza nel registro dei componenti sul dispositivo principale Greengrass e nel client AWS IoT MQTT nella console. AWS IoT Per sottoscrivere l'argomento su cui il componente pubblica i risultati dell'inferenza, vedere. Passaggio 1: iscriviti all'argomento delle notifiche predefinito

- AWS IoTClient MQTT: per visualizzare i risultati pubblicati dal componente di inferenza sull'argomento delle notifiche predefinite, completate i seguenti passaggi:

  1. Nel menu di navigazione della AWS IoTconsole, scegliete Test, MQTT test client.
  2. In Abbonamenti, scegli. **`ml/tflite/image-classification`**

     Dovresti vedere messaggi simili all'esempio seguente.

```
{
  "timestamp": "2021-01-01 00:00:00.000000",
  "inference-type": "image-classification",
  "inference-description": "Top 5 predictions with score 0.3 or above ",
  "inference-results": [
    {
      "Label": "cougar, puma, catamount, mountain lion, painter, panther, Felis
 concolor",
      "Score": "0.5882352941176471"
    },
    {
      "Label": "Persian cat",
      "Score": "0.5882352941176471"
    },
    {
      "Label": "tiger cat",
      "Score": "0.5882352941176471"
    },
    {
      "Label": "dalmatian, coach dog, carriage dog",
      "Score": "0.5607843137254902"
    },
    {
```

```
      "Label": "malamute, malemute, Alaskan malamute",
      "Score": "0.5450980392156862"
    }
  ]
}
```

- Registro dei componenti: per visualizzare i risultati dell'inferenza nel registro dei componenti, esegui il seguente comando sul tuo dispositivo principale Greengrass.

  ```
  sudo tail -f /greengrass/v2/logs/aws.greengrass.TensorFlowLiteImageClassification.log
  ```

  Dovreste vedere risultati simili a quelli dell'esempio seguente.

  ```
  2021-01-01 00:00:00.000000 [INFO] (Copier)
   aws.greengrass.TensorFlowLiteImageClassification: stdout. Publishing results to the
   IoT core....
   {scriptName=services.aws.greengrass.TensorFlowLiteImageClassification.lifecycle.Run.script,
   serviceName=aws.greengrass.TensorFlowLiteImageClassification, currentState=RUNNING}

  2021-01-01 00:00:00.000000 [INFO] (Copier)
   aws.greengrass.TensorFlowLiteImageClassification: stdout. {"timestamp":
   "2021-01-01 00:00:00.000000", "inference-type": "image-classification", "inference-
  description": "Top 5 predictions with score 0.3 or above ", "inference-results":
   [{"Label": "cougar, puma, catamount, mountain lion, painter, panther, Felis
   concolor", "Score": "0.5882352941176471"}, {"Label": "Persian cat", "Score":
   "0.5882352941176471"}, {"Label": "tiger cat", "Score": "0.5882352941176471"},
   {"Label": "dalmatian, coach dog, carriage dog", "Score": "0.5607843137254902"},
   {"Label": "malamute, malemute, Alaskan malamute", "Score": "0.5450980392156862"}]}.
   {scriptName=services.aws.greengrass.TensorFlowLiteImageClassification.lifecycle.Run.script,
   serviceName=aws.greengrass.TensorFlowLiteImageClassification, currentState=RUNNING}
  ```

Se non riesci a visualizzare i risultati dell'inferenza nel registro dei componenti o nel client MQTT, la distribuzione non è riuscita o non ha raggiunto il dispositivo principale. Ciò può verificarsi se il dispositivo principale non è connesso a Internet o non dispone delle autorizzazioni necessarie per eseguire il componente. Esegui il comando seguente sul tuo dispositivo principale per visualizzare il file di registro del software AWS IoT Greengrass Core. Questo file include i log del servizio di distribuzione del dispositivo principale Greengrass.

```
sudo tail -f /greengrass/v2/logs/greengrass.log
```

Per ulteriori informazioni, consulta Risoluzione dei problemi di inferenza dell'apprendimento automatico.

Passaggi successivi

Se disponi di un dispositivo principale Greengrass con un'interfaccia fotocamera supportata, puoi completareTutorial: Esegui l'inferenza della classificazione delle immagini di esempio sulle immagini di una fotocamera utilizzando TensorFlow Lite, che mostra come modificare il componente di inferenza del campione per eseguire la classificazione delle immagini sulle immagini di una fotocamera.

Per esplorare ulteriormente la configurazione del componente di inferenza per la classificazione delle immagini Sample TensorFlow Lite, provate quanto segue:

- Modifica il parametro `InferenceInterval` di configurazione per modificare la frequenza di esecuzione del codice di inferenza.
- Modifica i parametri `ImageName` e `ImageDirectory` di configurazione nella configurazione del componente di inferenza per specificare un'immagine personalizzata da utilizzare per l'inferenza.

Per informazioni sulla personalizzazione della configurazione dei componenti pubblici o sulla creazione di componenti di machine learning personalizzati, consulta. Personalizza i tuoi componenti di machine learning

Tutorial: Esegui l'inferenza della classificazione delle immagini di esempio sulle immagini di una fotocamera utilizzando TensorFlow Lite

Questo tutorial mostra come utilizzare il componente di inferenza della classificazione delle immagini TensorFlow Lite per eseguire l'inferenza della classificazione delle immagini di esempio sulle immagini di una fotocamera localmente su un dispositivo principale Greengrass. Questo componente include le seguenti dipendenze dei componenti:

- TensorFlow Componente di archiviazione del modello di classificazione delle immagini Lite
- TensorFlow Componente di runtime Lite

> **Note**
>
> Questo tutorial accede al modulo fotocamera per i dispositivi Raspberry Pi o NVIDIA Jetson Nano, ma AWS IoT Greengrass supporta altri dispositivi su piattaforme ARMv7L, Armv8 o x86_64. Per configurare una fotocamera per un altro dispositivo, consulta la documentazione pertinente del dispositivo.

Per ulteriori informazioni sull'apprendimento automatico sui dispositivi Greengrass, vedere. Esecuzione dell'inferenza di Machine Learning

Argomenti

Prerequisiti

Per completare questo tutorial, devi prima completareTutorial: eseguire l'inferenza della classificazione delle immagini di esempio utilizzando TensorFlow Lite.

Devi disporre anche dei seguenti elementi:

- Un dispositivo core Linux Greengrass con interfaccia fotocamera. Questo tutorial accede al modulo fotocamera su uno dei seguenti dispositivi supportati:
  - Raspberry Pi con sistema operativo Raspberry Pi (precedentemente chiamato Raspbian)
  - NVIDIA Jetson Nano

  Per informazioni sulla configurazione di un dispositivo principale Greengrass, vedere. Tutorial: Nozioni di base su AWS IoT Greengrass V2

  Il dispositivo principale deve soddisfare i seguenti requisiti:

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

    Per abilitare lo stack di telecamere precedente

    1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleziona Opzioni di interfaccia.
    3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
    4. Riavvia il dispositivo Raspberry Pi.
- Per dispositivi Raspberry Pi o NVIDIA Jetson Nano, modulo fotocamera Raspberry Pi V2 - 8 megapixel, 1080p. Per informazioni sulla configurazione della telecamera, consulta Connessione della telecamera nella documentazione di Raspberry Pi.

Passaggio 1: configura il modulo fotocamera sul tuo dispositivo

In questo passaggio, installi e abiliti il modulo videocamera per il tuo dispositivo. Esegui i seguenti comandi sul dispositivo.

Raspberry Pi (Armv7l)

1. Installa l'`picamera`interfaccia per il modulo videocamera. Esegui il seguente comando per installare il modulo telecamera e le altre librerie Python necessarie per questo tutorial.

   ```
   sudo apt-get install -y python3-picamera
   ```

2. Verificate che Picamera sia stato installato correttamente.

   ```
   sudo -u ggc_user bash -c 'python3 -c "import picamera"'
   ```

   Se l'output non contiene errori, la convalida ha esito positivo.

   > **Note**
   >
   > Se il file eseguibile Python installato sul tuo dispositivo lo è`python3.7`, usalo `python3.7` al posto dei comandi `python3` di questo tutorial. Assicurati che l'installazione di pip sia mappata alla versione `python3.7` o `python3` corretta per evitare errori di dipendenza.

3. Riavviare il dispositivo.

   ```
   sudo reboot
   ```

4. Apri lo strumento di configurazione di Raspberry Pi.

   ```
   sudo raspi-config
   ```

5. Utilizza i tasti freccia per aprire Interfacing Options (Opzioni di interfaccia) e abilita l'interfaccia della telecamera. Se richiesto, consenti il riavvio del dispositivo.

6. Eseguite il comando seguente per testare la configurazione della fotocamera.

   ```
   raspistill -v -o test.jpg
   ```

   Viene visualizzata una finestra di anteprima sul dispositivo Raspberry Pi, viene salvata un'immagine denominata `test.jpg` nella directory corrente e vengono visualizzati informazioni sulla telecamera nel terminale Raspberry Pi.

7. Eseguite il comando seguente per creare un collegamento simbolico che consenta al componente di inferenza di accedere alla telecamera dall'ambiente virtuale creato dal componente runtime.

   ```
   sudo ln -s /usr/lib/python3/dist-packages/picamera "MLRootPath/
   greengrass_ml_tflite_venv/lib/python3.7/site-packages"
   ```

   Il valore predefinito per *ML RootPath* per questo tutorial è. */greengrass/v2*/work/`variant.TensorFlowLite/greengrass_ml` La `greengrass_ml_tflite_venv` cartella in questa posizione viene creata quando si distribuisce il componente di inferenza per la prima volta in. Tutorial: eseguire l'inferenza della classificazione delle immagini di esempio utilizzando TensorFlow Lite

Jetson Nano (Armv8)

1. Eseguite il comando seguente per testare la configurazione della telecamera.

   ```
   gst-launch-1.0 nvarguscamerasrc num-buffers=1 ! "video/x-raw(memory:NVMM),
    width=1920, height=1080, format=NV12, framerate=30/1" ! nvjpegenc ! filesink
    location=test.jpg
   ```

   In questo modo viene acquisita e salvata un'immagine `test.jpg` denominata nella directory corrente.

2. (Facoltativo) Riavviare il dispositivo. Se riscontri problemi durante l'esecuzione del `gst-launch` comando nel passaggio precedente, il riavvio del dispositivo potrebbe risolverli.

   ```
   sudo reboot
   ```

> **Note**
>
> Per i dispositivi Armv8 (AArch64), come Jetson Nano, non è necessario creare un collegamento simbolico per consentire al componente di inferenza di accedere alla telecamera dall'ambiente virtuale creato dal componente runtime.

Passaggio 2: verifica l'iscrizione all'argomento delle notifiche predefinito

NelTutorial: eseguire l'inferenza della classificazione delle immagini di esempio utilizzando TensorFlow Lite, hai configurato il client AWS IoT MQTT è configurato nella AWS IoT console per guardare i messaggi MQTT pubblicati dal componente di classificazione delle immagini TensorFlow Lite sull'`ml/tflite/image-classification`argomento. Nella AWS IoT console, verifica che questo abbonamento esista. In caso contrario, segui i passaggi indicati Passaggio 1: iscriviti all'argomento delle notifiche predefinito per iscriverti a questo argomento prima di distribuire il componente sul tuo dispositivo principale Greengrass.

Passaggio 3: modifica la configurazione del componente di classificazione delle immagini TensorFlow Lite e distribuiscilo

In questo passaggio, configuri e distribuisci il componente di classificazione delle immagini TensorFlow Lite sul tuo dispositivo principale:

Per configurare e distribuire il componente di classificazione delle immagini TensorFlow Lite (console)

1. Nel menu di navigazione AWS IoT Greengrassdella console, scegli Componenti.
2. Nella pagina Componenti, nella scheda Componenti pubblici, scegli `aws.greengrass.TensorFlowLiteImageClassification`.
3. Nella pagina aws.greengrass.TensorFlowLiteImageClassification, scegli (Distribuisci).
4. Da Aggiungi alla distribuzione, scegli una delle seguenti opzioni:

    a. Per unire questo componente a una distribuzione esistente sul dispositivo di destinazione, scegli Aggiungi alla distribuzione esistente, quindi seleziona la distribuzione che desideri modificare.

    b. Per creare una nuova distribuzione sul dispositivo di destinazione, scegli Crea nuova distribuzione. Se hai una distribuzione esistente sul tuo dispositivo, la scelta di questo passaggio sostituisce la distribuzione esistente.
5. Nella pagina Specifica destinazione, procedi come segue:

    a. In Informazioni sulla distribuzione, inserisci o modifica il nome descrittivo della distribuzione.

    b. In Destinazione della distribuzione, seleziona una destinazione della distribuzione e scegli Avanti. Non è possibile modificare la destinazione della distribuzione se si sta revisionando una distribuzione esistente.

6. Nella pagina Seleziona componenti, in Componenti pubblici, verifica che il `aws.greengrass.TensorFlowLiteImageClassification` componente sia selezionato e scegli Avanti.
7. Nella pagina Configura componenti, procedi come segue:
   a. Selezionate il componente di inferenza e scegliete Configura componente.
   b. In Aggiornamento della configurazione, inserisci il seguente aggiornamento di configurazione nella casella Configurazione da unire.

      ```
      {
        "InferenceInterval": "60",
        "UseCamera": "true"
      }
      ```

      Con questo aggiornamento della configurazione, il componente accede al modulo della fotocamera sul dispositivo ed esegue inferenze sulle immagini scattate dalla fotocamera. Il codice di inferenza viene eseguito ogni 60 secondi.
   c. Seleziona Conferma e scegli Avanti.
8. Nella pagina Configura impostazioni avanzate, mantieni le impostazioni di configurazione predefinite e scegli Avanti.
9. Nella pagina di revisione, scegli Deploy

Per configurare e distribuire il componente di classificazione delle immagini TensorFlow Lite () AWS CLI

1. Crea un `deployment.json` file per definire la configurazione di distribuzione per il componente di classificazione delle immagini TensorFlow Lite. Questo file dovrebbe avere il seguente aspetto:

   ```
   {
     "targetArn":"targetArn",
     "components": {
       "aws.greengrass.TensorFlowLiteImageClassification": {
         "componentVersion": 2.1.0,
         "configurationUpdate": {
           "InferenceInterval": "60",
           "UseCamera": "true"
         }
   ```

```
    }
  }
}
```

- Nel campo `targetArn`, sostituisci *`targetArn`* con l'ARN (Amazon Resource Name) dell'oggetto o del gruppo di oggetti a cui destinare la distribuzione, nel seguente formato:
  - Oggetto: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
  - Gruppo di oggetti: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*
- Questo tutorial utilizza la versione 2.1.0 del componente. Nell'oggetto `aws.greengrass.TensorFlowLiteImageClassification` componente, sostituite *`2.1.0`* per utilizzare una versione diversa del componente di classificazione delle immagini TensorFlow Lite.

Con questo aggiornamento della configurazione, il componente accede al modulo fotocamera sul dispositivo ed esegue inferenze sulle immagini scattate dalla fotocamera. Il codice di inferenza viene eseguito ogni 60 secondi. Sostituisci i seguenti valori

2. Esegui il comando seguente per distribuire il componente di classificazione delle immagini TensorFlow Lite sul dispositivo:

```
aws greengrassv2 create-deployment \
    --cli-input-json file://path/to/deployment.json
```

La distribuzione può richiedere alcuni minuti. Nel passaggio successivo, controlla il registro dei componenti per verificare che la distribuzione sia stata completata correttamente e per visualizzare i risultati dell'inferenza.

Fase 4: Visualizzazione dei risultati dell'inferenza

Dopo aver distribuito il componente, è possibile visualizzare i risultati dell'inferenza nel registro dei componenti sul dispositivo principale Greengrass e nel client AWS IoT MQTT nella console. AWS IoT Per sottoscrivere l'argomento su cui il componente pubblica i risultati dell'inferenza, vedere. Passaggio 2: verifica l'iscrizione all'argomento delle notifiche predefinito

- AWS IoTClient MQTT: per visualizzare i risultati pubblicati dal componente di inferenza sull'argomento delle notifiche predefinite, completate i seguenti passaggi:

1. Nel menu di navigazione della AWS IoTconsole, scegliete Test, MQTT test client.
2. In Abbonamenti, scegli. **`ml/tflite/image-classification`**

- Registro dei componenti: per visualizzare i risultati dell'inferenza nel registro dei componenti, esegui il seguente comando sul tuo dispositivo principale Greengrass.

```
sudo tail -f /greengrass/v2/logs/aws.greengrass.TensorFlowLiteImageClassification.log
```

Se non riesci a visualizzare i risultati dell'inferenza nel registro dei componenti o nel client MQTT, la distribuzione non è riuscita o non ha raggiunto il dispositivo principale. Ciò può verificarsi se il dispositivo principale non è connesso a Internet o non dispone delle autorizzazioni necessarie per eseguire il componente. Esegui il comando seguente sul tuo dispositivo principale per visualizzare il file di registro del software AWS IoT Greengrass Core. Questo file include i log del servizio di distribuzione del dispositivo principale Greengrass.

```
sudo tail -f /greengrass/v2/logs/greengrass.log
```

Per ulteriori informazioni, consulta Risoluzione dei problemi di inferenza dell'apprendimento automatico.

Passaggi successivi

Questo tutorial mostra come utilizzare il componente di classificazione delle immagini TensorFlow Lite, con opzioni di configurazione personalizzate per eseguire la classificazione delle immagini di esempio sulle immagini scattate da una fotocamera.

Per ulteriori informazioni sulla personalizzazione della configurazione dei componenti pubblici o sulla creazione di componenti di machine learning personalizzati, consultaPersonalizza i tuoi componenti di machine learning.

Componenti

AWS IoT Greengrassi componenti sono moduli software che vengono distribuiti sui dispositivi core Greengrass. I componenti possono rappresentare applicazioni, programmi di installazione di runtime, librerie o qualsiasi codice da eseguire su un dispositivo. È possibile definire componenti che dipendono da altri componenti. Ad esempio, è possibile definire un componente che installa Python e quindi definire tale componente come dipendenza dei componenti che eseguono applicazioni Python. Quando distribuisci i componenti alle tue flotte di dispositivi, Greengrass implementa solo i moduli software richiesti dai tuoi dispositivi.

Argomenti

AWS-componenti forniti

AWS IoT Greengrass fornisce e gestisce componenti predefiniti che è possibile distribuire sui dispositivi. Questi componenti includono funzionalità (come lo stream manager), connettori AWS IoT Greengrass V1 (come le CloudWatch metriche) e strumenti di sviluppo locale (come la CLI AWS IoT Greengrass). Puoi distribuire questi componenti sui tuoi dispositivi per la loro funzionalità autonoma oppure puoi usarli come dipendenze nei componenti Greengrass personalizzati.

> **Note**
>
> Diversi componenti AWS forniti dipendono da versioni minori specifiche del nucleo Greengrass. A causa di questa dipendenza, è necessario aggiornare questi componenti quando si aggiorna il nucleo di Greengrass a una nuova versione secondaria. Per informazioni sulle versioni specifiche del nucleo da cui dipende ogni componente, consultate l'argomento relativo ai componenti. Per ulteriori informazioni sull'aggiornamento del nucleo, vedere. Aggiornamento del software AWS IoT Greengrass Core (OTA)

Quando un componente ha un tipo di componente sia generico che Lambda, la versione corrente del componente è di tipo generico e una versione precedente del componente è di tipo Lambda.

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Nucleo Greengrass	Il nucleo del software AWS IoT Greengrass Core. Usa questo componente per configurare e aggiornare il software sui tuoi dispositivi principali.	Nucleo	Linux, Windows	Sì
Autenticazione del dispositivo client	Consente ai dispositivi IoT locali, chiamati dispositivi client, di connettersi al dispositivo principale.	Plug-in	Linux, Windows	Sì
CloudWatch metriche	Pubblica metriche personalizzate su Amazon CloudWatch.	Generico, Lambda	Linux, Windows	Sì
AWS IoT Device Defender	Notifica agli amministratori le modifiche allo stato del	Generico, Lambda	Linux, Windows	Sì

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
	dispositivo principale Greengrass per identificare comportamenti insoliti.			
Spooler del disco	Abilita un'opzione di archiviazione persistente per i messaggi trasmessi dai dispositivi core Greengrass a. AWS IoT Core Questo componente memorizzerà questi messaggi in uscita su disco.	Plug-in	Linux, Windows	Sì

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Gestore di applicazioni Docker	Consente di AWS IoT Greengrass scaricare immagini Docker da Docker Hub e Amazon Elastic Container Registry (Amazon ECR).	Generico	Linux, Windows	No
Connettore Edge per Kinesis Video Streams	Legge i feed video dalle telecamere locali, pubblica gli stream su Kinesis Video Streams e li visualizza nelle dashboard Grafana con. AWS IoT TwinMaker	Generico	Linux	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Greengrass CLI	Fornisce un'interfaccia a riga di comando che è possibile utilizzare per creare distribuzioni locali e interagire con il dispositivo principale Greengrass e i suoi componenti.	Plug-in	Linux, Windows	Sì
Rilevatore IP	Riporta le informazioni sulla connettività del broker MQTT a AWS IoT Greengrass, in modo che i dispositivi client possano scoprire come connettersi.	Plug-in	Linux, Windows	Sì

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Firehose	Pubblica i dati tramite i flussi di distribuzione di Amazon Data Firehose verso destinazioni in. Cloud AWS	Lambda	Linux	No
Lanciatore Lambda	Gestisce i processi e la configurazione dell'ambiente per le funzioni Lambda.	Generico	Linux	No
Gestore Lambda	Gestisce la comunicazione tra processi e la scalabilità per le funzioni Lambda.	Plug-in	Linux	No
Runtime Lambda	Fornisce artefatti per ogni runtime Lambda.	Generico	Linux	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Router di abbonamento legacy	Gestisce gli abbonamenti per le funzioni Lambda eseguite AWS IoT Greengrass su V1.	Generico	Linux	No
Console di debug locale	Fornisce una console locale che è possibile utilizzare per eseguire il debug e gestire il dispositivo principale Greengrass e i suoi componenti.	Plug-in	Linux, Windows	Sì
Gestore dei registri	Raccoglie e carica i log sul dispositivo principale Greengrass.	Plug-in	Linux, Windows	Sì

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Componenti per l'apprendimento automatico	Fornisce modelli di apprendimento automatico e codice di inferenza di esempio che è possibile utilizzare per eseguire inferenze di apprendimento automatico sui dispositivi core Greengrass.	Per informazioni, consulta Componenti per l'apprendimento automatico.		
Adattatore di protocollo Modbus-RTU	Esamina le informazioni dai dispositivi Modbus RTU locali.	Lambda	Linux	No
Emettitore di telemetria Nucleus	Pubblica i dati di telemetria sanitaria del sistema raccolti dal nucleo su un argomento locale o su un argomento MQTT. AWS IoT Core	Plug-in	Linux, Windows	Sì

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Ponte MQTT	Inoltra messaggi MQTT tra dispositivi client, pubblicazione/sottoscrizione locale e. AWS IoT Greengrass AWS IoT Core	Plug-in	Linux, Windows	Sì
Broker MQTT 3.1.1 (Moquette)	Esegue un broker MQTT 3.1.1 che gestisce i messaggi tra i dispositivi client e il dispositivo principale.	Plug-in	Linux, Windows	Sì
Broker MQTT 5 (EMQX)	Esegue un broker MQTT 5 che gestisce i messaggi tra i dispositivi client e il dispositivo principale.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Fornitore PKCS #11	Consente ai componenti Greengrass di accedere a una chiave privata e a un certificato archiviati in modo sicuro in un modulo di sicurezza hardware (HSM).	Plug-in	Linux	Sì
Gestore segreto	Distribuisce i segreti dai AWS Secrets Manager segreti in modo da poter utilizzare in modo sicuro le credenziali, come le password, nei componenti personalizzati del dispositivo principale Greengrass.	Plug-in	Linux, Windows	Sì

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Tunneling sicuro	Consente connessioni di tunneling AWS IoT sicure che è possibile utilizzare per stabilire comunicazioni bidirezionali con i dispositivi core Greengrass protetti da firewall limitati.	Generico	Linux	No
Gestore delle ombre	Consente l'interazione con le ombre sul dispositivo principale. Gestisce l'archiviazione dei documenti shadow e anche la sincronizzazione degli stati shadow locali con il servizio AWS IoT Device Shadow.	Plug-in	Linux, Windows	Sì

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Amazon SNS	Pubblica messaggi su argomenti di Amazon SNS.	Lambda	Linux	No
Stream manager	Trasmette dati ad alto volume da fonti locali a. Cloud AWS	Generico	Linux, Windows	No
Agente Systems Manager	Gestisci il dispositivo principale con AWS Systems Manager, che ti consente di applicare patch ai dispositivi, eseguire comandi e altro ancora.	Generico	Linux	No
Servizio di scambio di token	Fornisce AWS credenziali che è possibile utilizzare per interagire con AWS i servizi.	Generico	Linux, Windows	No
Collettore IoT SiteWise OPC-UA	Raccoglie dati dai server OPC-UA.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Simulatore di sorgenti dati IoT SiteWise OPC-UA	Esegue un server OPC-UA locale che genera dati di esempio.	Generico	Linux, Windows	No
SiteWise Editore IoT	Pubblica dati nel AWS cloud.	Generico	Linux, Windows	No
SiteWise Processore IoT	Elabora i dati sui dispositivi principali Greengrass.	Generico	Linux, Windows	No

Nucleo Greengrass

Il componente Greengrass nucleus (`aws.greengrass.Nucleus`) è un componente obbligatorio e il requisito minimo per eseguire il software AWS IoT Greengrass Core su un dispositivo. È possibile configurare questo componente per personalizzare e aggiornare il software AWS IoT Greengrass Core da remoto. Implementa questo componente per configurare impostazioni come proxy, ruolo del dispositivo e configurazione degli AWS IoT oggetti sui tuoi dispositivi principali.

> **Important**
>
> Quando la versione del componente nucleus cambia o quando modifichi determinati parametri di configurazione, il software AWS IoT Greengrass Core, che include il nucleus e tutti gli altri componenti del dispositivo, si riavvia per applicare le modifiche.
> Quando distribuisci un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze del componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna

la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.12.x
- 2.11. x
- 2.10.x
- 2.9. x
- 2.8.x
- 2.7.x
- 2.6. x
- 2,5. x
- 2.4.x
- 2.3.x

- 2.2.x
- 2.1.x
- 2,0x

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Per ulteriori informazioni, consulta Piattaforme supportate.

Requisiti

I dispositivi devono soddisfare determinati requisiti per installare ed eseguire il Greengrass nucleus e il AWS IoT Greengrass software Core. Per ulteriori informazioni, consulta Requisiti per il dispositivo.

Il componente Greengrass nucleus è supportato per l'esecuzione in un VPC. Per distribuire questo componente in un VPC, è necessario quanto segue.

- Il componente Greengrass nucleus deve avere connettività a AWS IoT data, AWS IoT Credentials e Amazon S3.

Dipendenze

Il nucleo Greengrass non include alcuna dipendenza dai componenti. Tuttavia, diversi componenti AWS forniti includono il nucleo come dipendenza. Per ulteriori informazioni, consulta AWS-componenti forniti.

Per ulteriori informazioni sulle dipendenze dei componenti, consulta il riferimento alla ricetta dei componenti.

Download e installazione

Puoi scaricare un programma di installazione che configura il componente Greengrass nucleus sul tuo dispositivo. Questo programma di installazione configura il dispositivo come dispositivo principale

Greengrass. È possibile eseguire due tipi di installazione: un'installazione rapida che crea AWS le risorse necessarie all'utente o un'installazione manuale in cui AWS le risorse vengono create dall'utente. Per ulteriori informazioni, consulta Installare il software AWS IoT Greengrass Core..

Puoi anche seguire un tutorial per installare il nucleo Greengrass ed esplorare lo sviluppo dei componenti Greengrass. Per ulteriori informazioni, consulta Tutorial: Nozioni di base su AWS IoT Greengrass V2.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare quando si distribuisce il componente. Alcuni parametri richiedono il riavvio del software AWS IoT Greengrass Core per avere effetto. Per ulteriori informazioni sui motivi e sulle modalità di configurazione di questo componente, consultaConfigurare il software AWS IoT Greengrass Core.

`iotRoleAlias`

L'alias del AWS IoT ruolo che rimanda a un ruolo IAM per lo scambio di token. Il fornitore di AWS IoT credenziali assume questo ruolo per consentire al dispositivo principale Greengrass di interagire con i servizi. AWS Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

Quando si esegue il software AWS IoT Greengrass Core con l'`--provision true`opzione, il software fornisce un alias di ruolo e ne imposta il valore nel componente nucleo.

`interpolateComponentConfiguration`

(Facoltativo) È possibile abilitare il Greengrass nucleus per interpolare le variabili di ricetta dei componenti nelle configurazioni dei componenti e unire gli aggiornamenti di configurazione. Si consiglia di impostare questa opzione `true` in modo che il dispositivo principale possa eseguire componenti Greengrass che utilizzano variabili di ricetta nelle loro configurazioni.

Questa funzionalità è disponibile per la versione 2.6.0 e successive di questo componente.

Impostazione predefinita: `false`

`networkProxy`

(Facoltativo) Il proxy di rete da utilizzare per tutte le connessioni. Per ulteriori informazioni, consulta Connessione alla porta 443 o tramite un proxy di rete.

> **Important**
>
> Quando si implementa una modifica a questo parametro di configurazione, il software AWS IoT Greengrass Core si riavvia per rendere effettiva la modifica.

Questo oggetto contiene le seguenti informazioni:

`noProxyAddresses`

(Facoltativo) Un elenco separato da virgole di indirizzi IP o nomi host esenti dal proxy.

`proxy`

Il proxy a cui connettersi. Questo oggetto contiene le seguenti informazioni:

`url`

L'URL del server proxy nel formato`scheme://userinfo@host:port`.

- `scheme`— Lo schema, che deve essere `http` o`https`.

  > **Important**
  >
  > I dispositivi core Greengrass devono eseguire Greengrass nucleus v2.5.0 o versione successiva per utilizzare i proxy HTTPS.
  > Se configuri un proxy HTTPS, devi aggiungere il certificato CA del server proxy al certificato Amazon root CA del dispositivo principale. Per ulteriori informazioni, consulta Abilita il dispositivo principale in modo che consideri attendibile un proxy HTTPS.

- `userinfo`— (Facoltativo) Le informazioni sul nome utente e sulla password. Se si specificano queste informazioni nel`url`, il dispositivo principale Greengrass ignora i `username` campi and. `password`
- `host`— Il nome host o l'indirizzo IP del server proxy.
- `port`— (Facoltativo) Il numero di porta. Se non specifichi la porta, il dispositivo principale Greengrass utilizza i seguenti valori predefiniti:
  - `http`— 80
  - `https`— 443

`username`

(Facoltativo) Il nome utente che autentica il server proxy.

`password`

(Facoltativo) La password che autentica il server proxy.

`mqtt`

(Opzionale) La configurazione MQTT per il dispositivo principale Greengrass. Per ulteriori informazioni, consulta Connessione alla porta 443 o tramite un proxy di rete.

> **Important**
>
> Quando si implementa una modifica a questo parametro di configurazione, il software AWS IoT Greengrass Core si riavvia per rendere effettiva la modifica.

Questo oggetto contiene le seguenti informazioni:

`port`

(Facoltativo) La porta da utilizzare per le connessioni MQTT.

Impostazione predefinita: `8883`

`keepAliveTimeoutMs`

(Facoltativo) La quantità di tempo in millisecondi tra ogni `PING` messaggio inviato dal client per mantenere attiva la connessione MQTT. Questo valore deve essere maggiore di. `pingTimeoutMs`

Impostazione predefinita: `60000` (60 secondi)

`pingTimeoutMs`

(Facoltativo) La quantità di tempo in millisecondi che il client attende di ricevere un `PINGACK` messaggio dal server. Se l'attesa supera il timeout, il dispositivo principale si chiude e riapre la connessione MQTT. Questo valore deve essere inferiore a. `keepAliveTimeoutMs`

Impostazione predefinita: `30000` (30 secondi)

`operationTimeoutMs`

(Facoltativo) La quantità di tempo in millisecondi che il client attende il completamento delle operazioni MQTT (come `CONNECT` o). `PUBLISH` Questa opzione non si applica ai messaggi `PING` MQTT o keep alive.

Impostazione predefinita: `30000` (30 secondi)

`maxInFlightPublishes`

(Facoltativo) Il numero massimo di messaggi MQTT QoS 1 non riconosciuti che possono essere in transito contemporaneamente.

Questa funzionalità è disponibile per la versione 2.1.0 e successive di questo componente.

Impostazione predefinita: `5`

Intervallo valido: valore massimo di 100

`maxMessageSizeInBytes`

(Facoltativo) La dimensione massima di un messaggio MQTT. Se un messaggio supera questa dimensione, il nucleo Greengrass lo respinge con un errore.

Questa funzionalità è disponibile per la versione 2.1.0 e successive di questo componente.

Predefinito: `131072` (128 KB)

Intervallo valido: valore massimo di `2621440` (2,5 MB)

`maxPublishRetry`

(Facoltativo) Il numero massimo di volte in cui riprovare un messaggio che non viene pubblicato. È possibile specificare di `-1` riprovare un numero illimitato di volte.

Questa funzionalità è disponibile per la versione 2.1.0 e successive di questo componente.

Impostazione predefinita: `100`

`spooler`

(Opzionale) La configurazione dello spooler MQTT per il dispositivo principale Greengrass. Questo oggetto contiene le seguenti informazioni:

`storageType`

Il tipo di archiviazione per l'archiviazione dei messaggi. Se `storageType` è impostato su`Disk`, `pluginName` può essere configurato. È possibile specificare `Memory` o `Disk`.

Questa funzionalità è disponibile per la versione 2.11.0 e successive del componente Greengrass nucleus.

> **Important**
>
> Se lo spooler MQTT `storageType` è impostato su `Disk` e si desidera effettuare il downgrade di Greengrass nucleus dalla versione 2.11.x a una versione precedente, è necessario modificare nuovamente la configurazione in. `Memory` L'unica configurazione `storageType` supportata nelle versioni 2.10.x e precedenti di Greengrass nucleus è. `Memory` La mancata osservanza di queste indicazioni può causare la rottura dello spooler. Ciò impedirebbe al dispositivo principale Greengrass di inviare messaggi MQTT a. Cloud AWS

Impostazione predefinita: `Memory`

`pluginName`

(Facoltativo) Il nome del componente del plugin. Questo componente verrà utilizzato solo se `storageType` è impostato su`Disk`. Questa opzione è predefinita `aws.greengrass.DiskSpooler` e utilizzerà quella fornita da GreenGrass. Spooler del disco

Questa funzionalità è disponibile per la versione 2.11.0 e successive del componente Greengrass nucleus.

Impostazione predefinita: `"aws.greengrass.DiskSpooler"`

`maxSizeInBytes`

(Facoltativo) La dimensione massima della cache in cui il dispositivo principale archivia i messaggi MQTT non elaborati in memoria. Se la cache è piena, i nuovi messaggi vengono rifiutati.

Impostazione predefinita: `2621440` (2,5 MB)

`keepQos0WhenOffline`

(Facoltativo) È possibile eseguire lo spool dei messaggi MQTT QoS 0 che il dispositivo principale riceve mentre è offline. Se imposti questa opzione su`true`, il dispositivo principale sposta i messaggi QoS 0 che non può inviare mentre è offline. Se imposti questa opzione su`false`, il dispositivo principale scarta questi messaggi. Il dispositivo principale esegue sempre lo spoiler dei messaggi QoS 1 a meno che lo spool non sia pieno.

Impostazione predefinita: `false`

`version`

(Opzionale) La versione di MQTT. È possibile specificare `mqtt3` o `mqtt5`.

Questa funzionalità è disponibile per la versione 2.10.0 e successive del componente Greengrass nucleus.

Impostazione predefinita: `mqtt5`

`receiveMaximum`

(Facoltativo) Il numero massimo di pacchetti QoS1 non riconosciuti che il broker può inviare.

Questa funzionalità è disponibile per la versione 2.10.0 e successive del componente Greengrass nucleus.

Impostazione predefinita: `100`

`sessionExpirySeconds`

(Facoltativo) La quantità di tempo in secondi che puoi richiedere per la durata di una sessione da IoT Core. L'impostazione predefinita è il tempo massimo supportato da AWS IoT Core.

Questa funzionalità è disponibile per la versione 2.10.0 e successive del componente Greengrass nucleus.

Impostazione predefinita: `604800 (7 days)`

`minimumReconnectDelaySeconds`

(Facoltativo) Un'opzione per il comportamento di riconnessione. Il tempo minimo in secondi per la riconnessione di MQTT.

Questa funzionalità è disponibile per la versione 2.10.0 e successive del componente Greengrass nucleus.

Impostazione predefinita: 1

`maximumReconnectDelaySeconds`

(Facoltativo) Un'opzione per il comportamento di riconnessione. Il tempo massimo in secondi per la riconnessione di MQTT.

Questa funzionalità è disponibile per la versione 2.10.0 e successive del componente Greengrass nucleus.

Impostazione predefinita: `120`

`minimumConnectedTimeBeforeRetryResetSeconds`

(Facoltativo) Un'opzione per il comportamento di riconnessione. Il periodo di tempo in secondi in cui una connessione deve essere attiva prima che il ritardo tra i tentativi venga ripristinato al minimo.

Questa funzionalità è disponibile per la versione 2.10.0 e successive del componente Greengrass nucleus.

Impostazione predefinita: `30`

`jvmOptions`

(Facoltativo) Le opzioni JVM da utilizzare per eseguire il software Core. AWS IoT Greengrass Per informazioni sulle opzioni JVM consigliate per l'esecuzione del software AWS IoT Greengrass Core, vedere. Controlla l'allocazione della memoria con le opzioni JVM

> **Important**
>
> Quando si implementa una modifica a questo parametro di configurazione, il software AWS IoT Greengrass Core si riavvia per rendere effettiva la modifica.

`iotDataEndpoint`

L'endpoint di AWS IoT dati per il tuo. Account AWS

Quando esegui il software AWS IoT Greengrass Core con l'`--provision true`opzione, il software ottiene i dati e le credenziali dagli endpoint AWS IoT e li imposta nel componente nucleus.

`iotCredEndpoint`

L'endpoint delle AWS IoT credenziali per il tuo. Account AWS

Quando esegui il software AWS IoT Greengrass Core con l'`--provision true`opzione, il software ottiene i dati e le credenziali dagli endpoint AWS IoT e li imposta nel componente nucleus.

`greengrassDataPlaneEndpoint`

Questa funzionalità è disponibile nella versione 2.7.0 e successive di questo componente.

Per ulteriori informazioni, consulta Utilizza un certificato del dispositivo firmato da una CA privata.

`greengrassDataPlanePort`

Questa funzionalità è disponibile nella versione 2.0.4 e successive di questo componente.

(Facoltativo) La porta da utilizzare per le connessioni sul piano dati. Per ulteriori informazioni, consulta Connessione alla porta 443 o tramite un proxy di rete.

> **Important**
>
> È necessario specificare una porta in cui il dispositivo può effettuare connessioni in uscita. Se si specifica una porta bloccata, il dispositivo non sarà in grado di connettersi per AWS IoT Greengrass ricevere distribuzioni.

Seleziona una delle opzioni seguenti:

- `443`
- `8443`

Impostazione predefinita: `8443`

`awsRegion`

Quello da usare Regione AWS .

`runWithDefault`

L'utente del sistema da utilizzare per eseguire i componenti.

> **Important**
>
> Quando si implementa una modifica a questo parametro di configurazione, il software AWS IoT Greengrass Core si riavvia per rendere effettiva la modifica.

Questo oggetto contiene le seguenti informazioni:

`posixUser`

Il nome o l'ID dell'utente del sistema e, facoltativamente, del gruppo di sistema utilizzato dal dispositivo principale per eseguire componenti generici e Lambda. Specifica l'utente e il gruppo separati da due punti (`:`) nel seguente formato: `user:group`. Il gruppo è facoltativo. Se non si specifica un gruppo, il software AWS IoT Greengrass Core utilizza il gruppo principale per l'utente. Ad esempio, puoi specificare `ggc_user` o `ggc_user:ggc_group`. Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti.

Quando si esegue il programma di installazione del software AWS IoT Greengrass Core con l'`--component-default-user` *`ggc_user`*`:`*`ggc_group`*opzione, il software imposta questo parametro nel componente nucleus.

`windowsUser`

Questa funzionalità è disponibile nella versione 2.5.0 e successive di questo componente.

Il nome dell'utente Windows da utilizzare per eseguire questo componente sui dispositivi Windows principali. L'utente deve esistere su ogni dispositivo Windows principale e il nome e la password devono essere memorizzati nell'istanza di Credentials Manager dell' LocalSystem account. Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti.

Quando si esegue il programma di installazione del software AWS IoT Greengrass Core con l'`--component-default-user` *`ggc_user`*opzione, il software imposta questo parametro nel componente nucleus.

`systemResourceLimits`

Questa funzionalità è disponibile nella versione 2.4.0 e successive di questo componente. AWS IoT Greengrass attualmente non supporta questa funzionalità sui dispositivi Windows core.

I limiti delle risorse di sistema da applicare ai processi dei componenti Lambda generici e non containerizzati per impostazione predefinita. È possibile ignorare i limiti delle risorse di sistema

per i singoli componenti quando si crea una distribuzione. Per ulteriori informazioni, consulta Configura i limiti delle risorse di sistema per i componenti.

Questo oggetto contiene le seguenti informazioni:

`cpus`

La quantità massima di tempo di CPU che i processi di ciascun componente possono utilizzare sul dispositivo principale. Il tempo totale della CPU di un dispositivo principale è equivalente al numero di core CPU del dispositivo. Ad esempio, su un dispositivo principale con 4 core CPU, puoi impostare questo valore in modo da 2 limitare i processi di ciascun componente al 50% di utilizzo di ciascun core della CPU. Su un dispositivo con 1 core di CPU, puoi impostare questo valore su `0.25` per limitare i processi di ciascun componente al 25 percento di utilizzo della CPU. Se imposti questo valore su un numero maggiore del numero di core della CPU, il software AWS IoT Greengrass Core non limita l'utilizzo della CPU da parte dei componenti.

`memory`

La quantità massima di RAM (in kilobyte) che i processi di ciascun componente possono utilizzare sul dispositivo principale.

`s3EndpointType`

(Facoltativo) Il tipo di endpoint S3. Questo parametro avrà effetto solo per la regione Stati Uniti orientali (Virginia settentrionale) (`us-east-1`). L'impostazione di questo parametro da qualsiasi altra regione verrà ignorata. Seleziona una delle opzioni seguenti:

- `REGIONAL`— Il client S3 e l'URL predefinito utilizzano l'endpoint regionale.
- `GLOBAL`— Il client S3 e l'URL predefinito utilizzano l'endpoint legacy.

Impostazione predefinita: `GLOBAL`

`logging`

(Facoltativo) La configurazione di registrazione per il dispositivo principale. Per ulteriori informazioni su come configurare e utilizzare i log di Greengrass, vedere. Monitora AWS IoT Greengrass i registri

Questo oggetto contiene le seguenti informazioni:

`level`

(Facoltativo) Il livello minimo di messaggi di registro da emettere.

Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`format`

(Facoltativo) Il formato dei dati dei log. Seleziona una delle opzioni seguenti:

- `TEXT`— Scegliete questa opzione se desiderate visualizzare i log sotto forma di testo.
- `JSON`— Scegliete questa opzione se desiderate visualizzare i log con il comando Greengrass CLI logs o interagire con i log a livello di codice.

Impostazione predefinita: `TEXT`

`outputType`

(Facoltativo) Il tipo di output per i log. Seleziona una delle opzioni seguenti:

- `FILE`— Il software AWS IoT Greengrass Core invia i log nei file nella directory specificata. `outputDirectory`
- `CONSOLE`— Il software AWS IoT Greengrass Core stampa i registri su. `stdout` Scegliete questa opzione per visualizzare i registri man mano che il dispositivo principale li stampa.

Impostazione predefinita: `FILE`

`fileSizeKB`

(Facoltativo) La dimensione massima di ogni file di registro (in kilobyte). Quando un file di registro supera questa dimensione massima, il software AWS IoT Greengrass Core crea un nuovo file di registro.

Questo parametro si applica solo quando si specifica `FILE` per`outputType`.

Impostazione predefinita: `1024`

`totalLogsSizeKB`

(Facoltativo) La dimensione totale massima dei file di registro (in kilobyte) per ogni componente, incluso il nucleo Greengrass. I file di registro del nucleo di Greengrass includono

anche i log dei componenti del plugin. Dopo che la dimensione totale dei file di registro di un componente supera questa dimensione massima, il software AWS IoT Greengrass Core elimina i file di registro più vecchi del componente.

Questo parametro è equivalente al parametro limite di spazio su disco (`diskSpaceLimit`) del componente di gestione dei registri, che è possibile specificare per il nucleo di Greengrass (sistema) e ogni componente. Il software AWS IoT Greengrass Core utilizza il minimo dei due valori come dimensione massima totale del log per il nucleo Greengrass e ciascun componente.

Questo parametro si applica solo quando si specifica `FILE` per. `outputType`

Impostazione predefinita: `10240`

`outputDirectory`

(Facoltativo) La directory di output per i file di registro.

Questo parametro si applica solo quando si specifica `FILE` per`outputType`.

Impostazione predefinita:*`/greengrass/v2`*`/logs`, *`/greengrass/v2`* dov'è la cartella AWS IoT Greengrass principale.

`fleetstatus`

Questo parametro è disponibile nella versione 2.1.0 e successive di questo componente.

(Facoltativo) La configurazione dello stato del parco macchine per il dispositivo principale.

Questo oggetto contiene le seguenti informazioni:

`periodicStatusPublishIntervalSeconds`

(Facoltativo) Il periodo di tempo (in secondi) tra il quale il dispositivo principale pubblica lo stato del dispositivo su. Cloud AWS

Minimo: `86400` (24 ore)

Impostazione predefinita: `86400` (24 ore)

`telemetry`

(Facoltativo) La configurazione della telemetria sanitaria del sistema per il dispositivo principale. Per ulteriori informazioni sulle metriche di telemetria e su come agire sui dati di telemetria,

consulta. Raccogli dati di telemetria sanitaria del sistema dai dispositivi principali AWS IoT Greengrass

Questo oggetto contiene le seguenti informazioni:

`enabled`

(Facoltativo) È possibile abilitare o disabilitare la telemetria.

Impostazione predefinita: `true`

`periodicAggregateMetricsIntervalSeconds`

(Facoltativo) L'intervallo (in secondi) durante il quale il dispositivo principale aggrega le metriche.

Se impostate questo valore su un valore inferiore al valore minimo supportato, il nucleo utilizza invece il valore predefinito.

Minimo: `3600`

Impostazione predefinita: `3600`

`periodicPublishMetricsIntervalSeconds`

(Facoltativo) Il periodo di tempo (in secondi) tra il quale il dispositivo principale pubblica le metriche di telemetria su. Cloud AWS

Se impostate questo valore su un valore inferiore al valore minimo supportato, il nucleo utilizza invece il valore predefinito.

Minimo: `86400`

Impostazione predefinita: `86400`

`deploymentPollingFrequencySeconds`

(Facoltativo) Il periodo in secondi in cui eseguire il sondaggio per le notifiche di distribuzione.

Impostazione predefinita: `15`

`componentStoreMaxSizeBytes`

(Facoltativo) La dimensione massima su disco dell'archivio componenti, che comprende le ricette e gli artefatti dei componenti.

Impostazione predefinita: `10000000000` (10 GB)

`platformOverride`

(Facoltativo) Un dizionario di attributi che identifica la piattaforma del dispositivo principale. Utilizzatelo per definire gli attributi personalizzati della piattaforma che le ricette dei componenti possono utilizzare per identificare il ciclo di vita e gli artefatti corretti per il componente. Ad esempio, è possibile definire un attributo di capacità hardware per distribuire solo il set minimo di artefatti per l'esecuzione di un componente. Per ulteriori informazioni, vedete il parametro manifest platform nella ricetta del componente.

È inoltre possibile utilizzare questo parametro per sovrascrivere gli attributi `os` e di `architecture` piattaforma del dispositivo principale.

`httpClient`

Questo parametro è disponibile nella versione 2.5.0 e successive di questo componente.

(Facoltativo) La configurazione del client HTTP per il dispositivo principale. Queste opzioni di configurazione si applicano a tutte le richieste HTTP effettuate da questo componente. Se un dispositivo principale funziona su una rete più lenta, puoi aumentare queste durate di timeout per evitare il timeout delle richieste HTTP.

Questo oggetto contiene le seguenti informazioni:

`connectionTimeoutMs`

(Facoltativo) La quantità di tempo (in millisecondi) di attesa dell'apertura di una connessione prima del timeout della richiesta di connessione.

Impostazione predefinita: `2000` (2 secondi)

`socketTimeoutMs`

(Facoltativo) La quantità di tempo (in millisecondi) di attesa per il trasferimento dei dati su una connessione aperta prima del timeout della connessione.

Impostazione predefinita: `30000` (30 secondi)

Example Esempio: fusione e aggiornamento della configurazione

```
{
  "iotRoleAlias": "GreengrassCoreTokenExchangeRoleAlias",
  "networkProxy": {
    "noProxyAddresses": "http://192.168.0.1,www.example.com",
    "proxy": {
```

```
      "url": "http://my-proxy-server:1100",
      "username": "Mary_Major",
      "password": "pass@word1357"
    }
  },
  "mqtt": {
    "port": 443
  },
  "greengrassDataPlanePort": 443,
  "jvmOptions": "-Xmx64m",
  "runWithDefault": {
    "posixUser": "ggc_user:ggc_group"
  }
}
```

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
2.12.4	Correzioni di bug e miglioramenti • Risolve un problema per cui il nucleo entra in una condizione di stallo durante l'avvio su alcuni dispositivi Linux.
2.12.3	**Warning** Questa versione non è più disponibile. I miglioramenti di questa versione sono disponibili nelle versioni successive di questo componente. Correzioni di bug e miglioramenti • Risolve un problema per cui il nucleo non riporta lo stato corretto del componente dopo il riavvio del nucleo e durante il ripristino del componente. • Correzioni di bug generali e miglioramenti.
2.12.2	Correzioni di bug e miglioramenti • Risolve un problema per cui i vecchi log non venivano puliti correttam ente. • Correzioni di bug generali e miglioramenti.
2.12.1	Correzioni di bug e miglioramenti • Risolve un problema a causa del quale il Nucleus poteva duplicare gli abbonamenti MQTT ad argomenti di distribuzione, con conseguente ulteriore registrazione e pubblicazione di file MQTT.
2.12.0	Nuove funzionalità • Consente di eseguire le fasi del ciclo di vita di bootstrap come parte di una distribuzione di rollback.

Versione	Modifiche
2.11.3	Correzioni di bug e miglioramenti • Risolve un problema nel nucleo a causa del quale poteva avviare erroneamente un componente quando le sue dipendenze non funzionav ano. Nuove funzionalità • Aggiunge un tipo di endpoint s3 configurabile.
2.11.2	Correzioni di bug e miglioramenti • Risolve un problema nel client Nucleus MQTT 5 che poteva apparire offline quando è in uso un numero elevato (> 50) di abbonamenti. • Aggiunge un nuovo tentativo per l'errore TCP del docker dial.
2.11.1	Correzioni di bug e miglioramenti • Risolve un problema per cui il nucleo non si avvia se un'attività di bootstrap fallisce e il file dei metadati di distribuzione è danneggiato. • Risolve un problema per cui i componenti Lambda su richiesta non vengono segnalati negli aggiornamenti sullo stato della distribuzione. • Aggiunge il supporto per gli ID duplicati delle politiche di autorizzazione.
2.11.0	Nuove funzionalità • Consente di annullare una distribuzione locale. • Consente di configurare una politica di gestione degli errori per una distribuzione locale. • Aggiunge il supporto per un plug-in Disk Spooler.
2.10.3	Correzioni di bug e miglioramenti • Risolve un problema per cui Greengrass non sottoscrive le notifiche di distribuzione quando utilizza il provider PKCS #11.

Versione	Modifiche
2.10.2	Correzioni di bug e miglioramenti • Consente l'analisi senza distinzione tra maiuscole e minuscole dei cicli di vita dei componenti. • Risolve un problema per cui la variabile di ambiente PATH non veniva ricreata correttamente. • Risolve la codifica URI del proxy per i componenti, incluso lo stream manager per i nomi utente con caratteri speciali.
2.10.1	Correzioni di bug e miglioramenti • Risolve un problema che poteva causare un arresto anomalo all'avvio su alcuni processori ARMv8, incluso Jetson Nano. • Greengrass non chiude più lo standard di un componente, ma ripristina il comportamento al comportamento precedente alla versione 2.10.0

Versione	Modifiche
2.10.0	Nuove funzionalità • Aggiunge il `interpolateComponentConfiguration` supporto per l'espressione regolare vuota. Greengrass ora esegue l'interpola dall'oggetto root config. • Aggiunge il supporto per MQTT5. • Aggiunge un meccanismo per caricare rapidamente i componenti del plug-in senza scansione. • Consente a Greengrass di risparmiare spazio su disco eliminando le immagini Docker inutilizzate. Correzioni di bug e miglioramenti • Risolve un problema per cui il rollback lascia determinati valori di configurazione al loro posto da una distribuzione. • Risolve un problema a causa del quale il nucleo Greengrass esegue la convalida di AWS una sequenza di dominio in endpoint di dati e non credenziali AWS personalizzati. • Aggiorna la risoluzione delle dipendenze multigruppo per risolvere nuovamente tutte le dipendenze di gruppo tramite Cloud AWS negoziazione, anziché limitarsi alla versione attiva. Questo aggiornam ento rimuove anche il codice di errore di distribuzione. `INSTALLED_COMPONENT_NOT_FOUND` • Aggiorna il nucleo Greengrass per saltare il download delle immagini Docker quando sono già presenti localmente. • Aggiorna il nucleo Greengrass per riavviare una fase di installazione dei componenti prima della scadenza del timeout. • Correzioni e miglioramenti minori aggiuntivi.
2.9.6	Correzioni di bug e miglioramenti • Risolve un problema per cui una distribuzione Greengrass fallisce con l'errore LAUNCH_DIRECTORY_CORRUPTED e un successivo riavvio del dispositivo non riesce ad avviare Greengrass. Questo errore può verificarsi quando si sposta il dispositivo Greengrass tra più gruppi di oggetti con distribuzioni che richiedono il riavvio di Greengrass.

Versione	Modifiche
2.9.5	Nuove funzionalità • Aggiunge il supporto per la verifica della firma del software Greengrass nucleus. Correzioni di bug e miglioramenti • Risolve un problema a causa del quale una distribuzione fallisce quando l'area dei metadati della ricetta locale non corrisponde alla regione di lancio di Greengrass nucleus. Il nucleo di Greengrass ora rinegozia con il cloud quando ciò accade. • Risolve un problema per cui lo spooler dei messaggi MQTT si riempie e non rimuove mai i messaggi. • Correzioni e miglioramenti minori aggiuntivi.
2.9.4	Correzioni di bug e miglioramenti • Verifica la presenza di un messaggio nullo prima di eliminare i messaggi QOS 0. • Tronca i valori di dettaglio dello stato del lavoro se superano il limite di 1024 caratteri. • Aggiorna lo script di bootstrap per Windows per leggere correttamente il percorso principale di Greengrass se tale percorso include spazi. • Aggiorna la sottoscrizione AWS IoT Core in modo da eliminare i messaggi del client se la risposta all'abbonamento non è stata inviata. • Assicura che il nucleo carichi la configurazione dai file di backup quando il file di configurazione principale è danneggiato o mancante.
2.9.3	Correzioni di bug e miglioramenti • Assicura che gli ID client MQTT non vengano duplicati. • Aggiunge funzionalità di lettura e scrittura dei file più affidabili per evitare e ripristinare eventuali danneggiamenti. • Riprova il docker image pull per errori specifici relativi alla rete. • Aggiunge l'opzione per la connessione MQTT`noProxyAddresses` .

Versione	Modifiche
2.9.2	Correzioni di bug e miglioramenti • Risolve un problema per cui la configurazione `interpolateComponentConfiguration` non si applica a una distribuzione in corso. • Utilizza OSHI per elencare tutti i processi secondari.
2.9.1	Correzioni di bug e miglioramenti • Aggiunge una correzione in cui Greengrass si riavvia se una distribuzione rimuove un componente del plug-in.
2.9.0	Nuove funzionalità • Aggiunge la possibilità di creare sottodistribuzioni che riprovano le distribuzioni con un sottoinsieme più piccolo di dispositivi. Questa funzionalità crea un modo più efficiente per testare e risolvere le distribuzioni non riuscite. Correzioni di bug e miglioramenti • Migliora il supporto per i sistemi che non dispongono di `useraddgroupadd`, `eusermod`. • Correzioni e miglioramenti minori aggiuntivi.
2.8.1	Correzioni di bug e miglioramenti • Risolve un problema a causa del quale i codici di errore di distribuzione non venivano generati correttamente dagli errori dell'API Greengrass. • Risolve un problema per cui gli aggiornamenti dello stato del parco macchine inviano informazioni imprecise quando un componente raggiunge `ERROR0RED` uno stato durante un'implementazione. • Risolve un problema per cui le distribuzioni non potevano essere completate quando Greengrass aveva più di 50 abbonamenti esistenti.

Versione	Modifiche
2.8.0	Nuove funzionalità • Aggiorna il Greengrass nucleus per segnalare una risposta sullo stato di integrità dell'implementazione che include codici di errore dettaglia ti in caso di problemi durante la distribuzione dei componenti su un dispositivo principale. Per ulteriori informazioni, consulta Codici di errore di distribuzione dettagliati. • Aggiorna il nucleo Greengrass per segnalare una risposta sullo stato di salute dei componenti che include codici di errore dettagliati quando un componente entra nello `BROKEN` stato or. `ERRORED` Per ulteriori informazioni, consulta Codici di stato dettagliati dei componenti. • Espande i campi dei messaggi di stato per migliorare le informazioni sulla disponibilità del cloud per i dispositivi. • Migliora lo stato della flotta e la robustezza del servizio. Correzioni di bug e miglioramenti • Consente la reinstallazione di un componente danneggiato quando la sua configurazione cambia. • Risolve un problema per cui un riavvio del nucleo durante l'impleme ntazione di bootstrap causa il fallimento di una distribuzione. • Risolve un problema in Windows a causa del quale l'installazione non riesce quando un percorso root contiene spazi. • Risolve un problema per cui l'arresto di un componente durante una distribuzione utilizza lo script di spegnimento della nuova versione. • Vari miglioramenti allo spegnimento. • Correzioni e miglioramenti minori aggiuntivi.

Versione	Modifiche
2.7.0	Nuove funzionalità • Aggiorna il nucleo Greengrass per inviare aggiornamenti di stato al AWS IoT Greengrass cloud quando il dispositivo principale applica una distribuzione locale. • Aggiunge il supporto per i certificati client firmati da un'autorità di certificazione (CA) personalizzata, presso la quale la CA non è registrat a. AWS IoT Per utilizzare questa funzionalità, puoi impostare la nuova opzione `greengrassDataPlaneEndpoint` di configurazione `suiotdata`. Per ulteriori informazioni, consulta Utilizza un certificato del dispositivo firmato da una CA privata. Correzioni di bug e miglioramenti • Risolve un problema per cui il nucleo Greengrass ripristina un dispiegam ento in determinati scenari quando il nucleo viene fermato o riavviato. Il nucleo ora riprende il dispiegamento dopo il riavvio del nucleo. • Aggiorna il programma di installazione di Greengrass per rispettare l'`--start`argomento quando si specifica di configurare il software come servizio di sistema. • Aggiorna il comportamento di SubscribeToComponentUpdates impostare l'ID di distribuzione negli eventi in cui il nucleo ha aggiornato un componente. • Correzioni e miglioramenti minori aggiuntivi.

Versione	Modifiche
2.6.0	Nuove funzionalità • Aggiunge il supporto per le wildcard MQTT quando ci si abbona ad argomenti di pubblicazione/sottoscrizione locali. Per ulteriori informazioni, consulta Pubblicare/sottoscrivere messaggi locali e Subscribe ToTopic. • Aggiunge il supporto per le variabili di ricetta nelle configurazioni dei componenti, diverse dalla variabile di ricetta. *component_dependency_name* :configuration: *json_pointer* È possibile utilizzare queste variabili di ricette quando si definisce un componente DefaultConfiguration in una ricetta o quando si configura un componente in una distribuzione. Per abilitare questa funzionalità, imposta l'opzione interpolateComponentConfigurationdi configurazione sutrue. Per ulteriori informazioni, consulta Variabili di ricetta e Usa le variabili di ricetta negli aggiornamenti di fusione. • Aggiunge il supporto completo per la * wildcard nelle politiche di autorizzazione della comunicazione tra processi (IPC). È ora possibile specificare il * carattere in una stringa di risorse in modo che corrisponda a qualsiasi combinazione di caratteri. Per ulteriori informazioni, consulta Wildcard nelle politiche di autorizzazione. • Aggiunge il supporto per i componenti personalizzati per chiamare le operazioni IPC utilizzate dalla CLI di Greengrass. È possibile utilizzare queste operazioni IPC per gestire le distribuzioni locali, visualizzare i dettagli dei componenti e generare una password che è possibile utilizzare per accedere alla console di debug locale. Per ulteriori informazioni, consulta IPC: gestione delle distribuzioni e dei componenti locali. Correzioni di bug e miglioramenti • Risolve un problema per cui i componenti dipendenti non reagivano quando le loro dipendenze rigide si riavviavano o cambiavano stato in determinati scenari. • Migliora i messaggi di errore che il dispositivo principale segnala al servizio AWS IoT Greengrass cloud quando un'implementazione fallisce.

Versione	Modifiche
	• Risolve un problema per cui il nucleo di Greengrass applicava due volte il dispiegamento di un oggetto in determinati scenari al riavvio del nucleo. • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.
2.5.6	Nuove funzionalità • Aggiunge il supporto per i moduli di sicurezza hardware che utilizzan o chiavi ECC. È possibile utilizzare un modulo di sicurezza hardware (HSM) per archiviare in modo sicuro la chiave privata e il certificato del dispositivo. Per ulteriori informazioni, consulta Integrazione della sicurezza hardware. Correzioni di bug e miglioramenti • Risolve un problema per cui la distribuzione non viene mai completata quando si distribuisce un componente con uno script di installazione non funzionante in determinati scenari. • Migliora le prestazioni durante l'avvio. • Correzioni e miglioramenti minori aggiuntivi.

Versione	Modifiche
2.5.5	Nuove funzionalità • Aggiunge la variabile di `GG_ROOT_CA_PATH` ambiente per i componenti, in modo da poter accedere al certificato dell'autorità di certificazione (CA) principale nei componenti personalizzati. Correzioni di bug e miglioramenti • Aggiunge il supporto per i dispositivi Windows che utilizzano una lingua di visualizzazione diversa dall'inglese. • Aggiorna il modo in cui il nucleo Greengrass analizza gli argomenti booleani dell'installatore, in modo da poter specificare un argomento booleano senza un valore booleano per specificare un valore. `true` Ad esempio, ora è possibile specificare invece di installare con il provisioning automatico delle risorse. `--provision --provision true` • Risolve un problema per cui il dispositivo principale non segnalava il proprio stato al servizio AWS IoT Greengrass cloud dopo il provisioning in determinati scenari. • Correzioni e miglioramenti minori aggiuntivi.
2.5.4	Correzioni di bug e miglioramenti • Correzioni di bug generali e miglioramenti.
2.5.3	Nuove funzionalità • Aggiunge il supporto per l'integrazione della sicurezza hardware. È possibile utilizzare un modulo di sicurezza hardware (HSM) per archiviare in modo sicuro la chiave privata e il certificato del dispositivo. Per ulteriori informazioni, consulta Integrazione della sicurezza hardware. Correzioni di bug e miglioramenti • Risolve un problema con le eccezioni di runtime mentre il nucleo stabilisce connessioni MQTT con. AWS IoT Core
2.5.2	Correzioni di bug e miglioramenti • Risolve un problema a causa del quale, dopo l'aggiornamento del Greengrass Nucleus, il servizio Windows non si riavvia dopo l'arresto o il riavvio del dispositivo.

Versione	Modifiche
2.5.1	**Warning** Questa versione non è più disponibile. I miglioramenti di questa versione sono disponibili nelle versioni successive di questo componente. Correzioni di bug e miglioramenti • Aggiunge il supporto per le versioni a 32 bit di Java Runtime Environme nt (JRE) su Windows. • Modifica il comportamento di rimozione dei gruppi di oggetti per i dispositivi principali la cui AWS IoT politica non concede l'`greengras s:ListThingGroupsForCoreDevice` autorizzazione. Con questa versione, la distribuzione continua, registra un avviso e non rimuove i componenti quando si rimuove il dispositivo principale da un gruppo di oggetti. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi. • Risolve un problema con le variabili di ambiente di sistema che il nucleo Greengrass mette a disposizione dei processi dei componenti Greengrass. È ora possibile riavviare un componente affinché utilizzi le variabili di ambiente di sistema più recenti.

Versione	Modifiche
2.5.0	Nuove funzionalità • Aggiunge il supporto per i dispositivi principali che eseguono Windows. • Modifica il comportamento della rimozione dei gruppi di oggetti. Con questa versione, è possibile rimuovere un dispositivo principale da un gruppo di oggetti per disinstallarne i componenti nella distribuzione successiva. A seguito di questa modifica, la AWS IoT policy di un dispositivo principale deve disporre dell'`greengrass:ListThingGroupsForCoreDevice` autorizzazione. Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per effettuare il provisioning delle risorse, lo consente la AWS IoT politica predefini ta`greengrass:*` , che include questa autorizzazione. Per ulteriori informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass. • Aggiunge il supporto per le configurazioni proxy HTTPS. Per ulteriori informazioni, consulta Connessione alla porta 443 o tramite un proxy di rete. • Aggiunge il nuovo parametro `windowsUser` di configurazione. È possibile utilizzare questo parametro per specificare l'utente predefini to da utilizzare per eseguire i componenti su un dispositivo Windows principale. Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti. • Aggiunge le nuove opzioni di `httpClient` configurazione che è possibile utilizzare per personalizzare i timeout delle richieste HTTP per migliorare le prestazioni su reti lente. Per ulteriori informazioni, vedete il parametro di configurazione HttpClient. Correzioni di bug e miglioramenti • Risolve l'opzione del ciclo di vita di bootstrap per riavviare il dispositivo principale da un componente. • Aggiunge il supporto per i trattini nelle variabili di ricetta. • Corregge l'autorizzazione IPC per i componenti della funzione Lambda su richiesta.

Versione	Modifiche
	• Migliora i messaggi di registro e modifica i log non critici da un `DEBUG` livello `INFO` all'altro, quindi i log sono più utili. • Rimuove l'`iot:DescribeCertificate` autorizzazione dal ruolo di scambio di token predefinito creato dal nucleo Greengrass quando si installa il software AWS IoT Greengrass Core con provisioning automatico. Questa autorizzazione non viene utilizzata dal nucleo Greengrass. • Risolve un problema per cui lo script di provisioning automatico non richiede l'`iam:GetPolicy` autorizzazione se `iam:CreatePolicy` disponibile per la stessa politica. • Correzioni e miglioramenti minori aggiuntivi.

Versione	Modifiche
2.4.0	Nuove funzionalità • Aggiunge il supporto per i limiti delle risorse di sistema. È possibile configurare la quantità massima di utilizzo di CPU e RAM che i processi di ciascun componente possono utilizzare sul dispositivo principale. Per ulteriori informazioni, consulta Configura i limiti delle risorse di sistema per i componenti. • Aggiunge operazioni IPC per mettere in pausa e riprendere i component i. Per ulteriori informazioni, consulta PauseComponent e ResumeCom ponent. • Aggiunge il supporto per i plugin di provisioning. È possibile specificare un file JAR da eseguire durante l'installazione per fornire AWS le risorse necessarie per un dispositivo centrale Greengrass. Il nucleo Greengrass include un'interfaccia che è possibile implementare per sviluppare plugin di provisioning personalizzati. Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning personalizzato delle risorse. • Aggiunge l'`thing-name-policy` argomento opzionale al programma di installazione del software Core AWS IoT Greengrass . È possibile utilizzare questa opzione per specificare una AWS IoT politica esistente o personalizzata quando si installa il software AWS IoT Greengrass Core con il provisioning automatico delle risorse. Correzioni di bug e miglioramenti • Aggiorna la configurazione della registrazione all'avvio. Questo risolve un problema a causa del quale la configurazione di registrazione non veniva applicata all'avvio. • Aggiorna il collegamento simbolico del nucleus loader in modo che punti all'archivio dei componenti nella cartella principale di Greengras s durante l'installazione. Questo aggiornamento consente di eliminare il file JAR e altri artefatti del nucleo scaricati durante l'installazione del software Core. AWS IoT Greengrass • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.

Versione	Modifiche
2.3.0	Nuove funzionalità • Aggiunge il supporto per i documenti di configurazione della distribuz ione fino a 10 MB, rispetto ai 7 KB (per le distribuzioni destinate a oggetti) o 31 KB (per le distribuzioni destinate a gruppi di oggetti). Per utilizzare questa funzionalità, una AWS IoT policy del dispositivo principale deve consentire l'autorizzazione. `greengrass:GetDepl oymentConfiguration` Se hai utilizzato il programma di installaz ione del software AWS IoT Greengrass Core per effettuare il provision ing delle risorse, lo consente la AWS IoT policy del dispositivo principal `egreengrass:*` , che include questa autorizzazione. Per ulteriori informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass. • Aggiunge la variabile `iot:thingName` recipe. Puoi usare questa variabile di ricetta per ottenere il nome dell'elemento del dispositivo AWS IoT principale in una ricetta. Per ulteriori informazioni, consulta Variabili di ricetta. Correzioni di bug e miglioramenti • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.
2.2.0	Nuove funzionalità • Aggiunge operazioni IPC per la gestione locale dello shadow. Correzioni di bug e miglioramenti • Riduce le dimensioni del file JAR. • Riduce l'utilizzo della memoria. • Risolve i problemi per cui la configurazione del registro non veniva aggiornata in alcuni casi. • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.

Versione	Modifiche
2.1.0	Nuove funzionalità • Supporta il download di immagini Docker da archivi privati in Amazon ECR. • Aggiunge i seguenti parametri per personalizzare la configurazione MQTT sui dispositivi principali: • `maxInFlightPublishes` — Il numero massimo di messaggi MQTT QoS 1 non riconosciuti che possono essere in transito contemporaneamente. • `maxPublishRetry` — Il numero massimo di volte in cui riprovare un messaggio che non viene pubblicato. • Aggiunge il parametro di `fleetstatusservice` configurazione per configurare l'intervallo con cui il dispositivo principale pubblica lo stato del dispositivo su. Cloud AWS • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub. Correzioni di bug e miglioramenti • Risolve un problema che causava la duplicazione delle distribuzioni shadow al riavvio del nucleo. • Risolve un problema che causava l'arresto anomalo del nucleus quando rilevava un'eccezione di carico del servizio. • Migliora la risoluzione delle dipendenze dei componenti per evitare il fallimento di una distribuzione che include una dipendenza circolare. • Risolve un problema che impediva la ridistribuzione di un componente del plug-in se tale componente era stato precedentemente rimosso dal dispositivo principale. • Risolve un problema che causava l'impostazione della variabile di `HOME` ambiente nella */greengrass/v2* `/work` directory per i componenti Lambda o per i componenti eseguiti come root. La `HOME` variabile è ora impostata correttamente nella home directory dell'utente che esegue il componente.

Versione	Modifiche
	• Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.
2.0.5	Correzioni di bug e miglioramenti • Indirizza correttamente il traffico attraverso un proxy di rete configurato durante il download dei AWS componenti forniti. • Usa l'endpoint del piano dati Greengrass corretto nelle AWS regioni cinesi.

Versione	Modifiche
2.0.4	Nuove funzionalità • Abilita il traffico HTTPS sulla porta 443. È possibile utilizzare il nuovo parametro di `greengrassDataPlanePort` configurazione per la versione 2.0.4 del componente nucleus per configurare la comunicaz ione HTTPS in modo che viaggi sulla porta 443 anziché sulla porta predefinita 8443. Per ulteriori informazioni, consulta Configura HTTPS sulla porta 443. • Aggiunge la variabile di ricetta del percorso di lavoro. È possibile utilizzare questa variabile di ricetta per ottenere il percorso delle cartelle di lavoro dei componenti, che è possibile utilizzare per condividere file tra i componenti e le relative dipendenze. Per ulteriori informazioni, consulta la variabile di ricetta del percorso di lavoro. Correzioni di bug e miglioramenti • Impedisce la creazione della politica di ruolo dello scambio di token AWS Identity and Access Management (IAM) se esiste già una politica di ruolo. Come risultato di questa modifica, il programma di installazione ora richiede `iam:GetPolicy` e `sts:GetCallerIdentity` quando viene eseguito con`--provision true` . Per ulteriori informazioni, consulta Policy IAM minima per l'installatore per il provisioning delle risorse. • Gestisce correttamente l'annullamento di una distribuzione che non è stata ancora registrata correttamente. • Aggiorna la configurazione per rimuovere le voci più vecchie con timestamp più recenti durante il rollback di una distribuzione. • Correzioni e miglioramenti minori aggiuntivi. Per ulteriori informazioni, consulta le versioni su GitHub.
2.0.3	Versione iniziale.

Autenticazione del dispositivo client

Il componente di autenticazione del dispositivo client (`aws.greengrass.clientdevices.Auth`) autentica i dispositivi client e autorizza le azioni dei dispositivi client.

> **Note**
>
> I dispositivi client sono dispositivi IoT locali che si connettono a un dispositivo core Greengrass per inviare messaggi MQTT e dati da elaborare. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

Argomenti

Versioni

> **Note**
>
> La versione 2.3.0 di autenticazione dei dispositivi client è stata interrotta. Si consiglia vivamente di eseguire l'aggiornamento alla versione 2.3.1 o successiva di autenticazione del dispositivo client.

Questo componente ha le seguenti versioni:

- 2.4.x
- 2.3.x

- 2.2.x
- 2.1.x
- 2,0x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Il ruolo del servizio Greengrass deve essere associato all'utente Account AWS e consentire l'`iot:DescribeCertificate`autorizzazione.
- La AWS IoT politica del dispositivo principale deve consentire le seguenti autorizzazioni:
  - `greengrass:GetConnectivityInfo`, dove le risorse includono l'ARN del dispositivo principale che esegue questo componente
  - `greengrass:VerifyClientDeviceIoTCertificateAssociation`, dove le risorse includono l'Amazon Resource Name (ARN) di ogni dispositivo client che si connette al dispositivo principale
  - `greengrass:VerifyClientDeviceIdentity`

- `greengrass:PutCertificateAuthorities`
- `iot:Publish`, dove le risorse includono l'ARN del seguente argomento MQTT:
  - `$aws/things/coreDeviceThingName*-gci/shadow/get`
- `iot:Subscribe`, dove le risorse includono gli ARN dei seguenti filtri tematici MQTT:
  - `$aws/things/coreDeviceThingName*-gci/shadow/update/delta`
  - `$aws/things/coreDeviceThingName*-gci/shadow/get/accepted`
- `iot:Receive`, dove le risorse includono gli ARN dei seguenti argomenti MQTT:
  - `$aws/things/coreDeviceThingName*-gci/shadow/update/delta`
  - `$aws/things/coreDeviceThingName*-gci/shadow/get/accepted`

  Per ulteriori informazioni, consulta Policy AWS IoT per operazioni del piano dei dati e AWS IoTPolitica minima per supportare i dispositivi client.

- (Facoltativo) Per utilizzare l'autenticazione offline, il ruolo AWS Identity and Access Management (IAM) utilizzato dal AWS IoT Greengrass servizio deve contenere le seguenti autorizzazioni:
  - `greengrass:ListClientDevicesAssociatedWithCoreDevice`per consentire al dispositivo principale di elencare i client per l'autenticazione offline.
- Il componente di autenticazione del dispositivo client è supportato per l'esecuzione in un VPC. Per distribuire questo componente in un VPC, è necessario quanto segue.
  - Il componente di autenticazione del dispositivo client deve avere connettività a AWS IoT data, AWS IoT Credentials e Amazon S3.

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`iot.region.amazonaws.com`	443	Sì	Utilizzato per ottenere informazi

Endpoint	Porta	Richiesto	Descrizione
			oni sui AWS IoT certificati Thing.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.4.4

La tabella seguente elenca le dipendenze per la versione 2.4.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.6.0 <2.13.0	Flessibili

2.4.3

La tabella seguente elenca le dipendenze per la versione 2.4.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.6.0 <2.12.0	Flessibili

2.4.1 and 2.4.2

La tabella seguente elenca le dipendenze per le versioni 2.4.1 e 2.4.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.6.0 <2.11.0	Flessibili

2.3.0 – 2.4.0

La tabella seguente elenca le dipendenze per le versioni da 2.3.0 a 2.4.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.6.0 <2.10.0	Flessibili

2.3.0

La tabella seguente elenca le dipendenze per la versione 2.3.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.6.0 <2.10.0	Flessibili

2.2.3

La tabella seguente elenca le dipendenze per la versione 2.2.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.6.0 <=2.9.0	Flessibili

2.2.2

La tabella seguente elenca le dipendenze per la versione 2.2.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.6.0 <=2.8.0	Flessibili

2.2.1

La tabella seguente elenca le dipendenze per la versione 2.2.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.6.0 <2.8.0	Flessibili

2.2.0

La tabella seguente elenca le dipendenze per la versione 2.2.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.6.0 <2.7.0	Flessibili

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.7.0	Flessibili

2.0.4

La tabella seguente elenca le dipendenze per la versione 2.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.6.0	Flessibili

2.0.2 and 2.0.3

La tabella seguente elenca le dipendenze per le versioni 2.0.2 e 2.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.5.0	Flessibili

2.0.1

La tabella seguente elenca le dipendenze per la versione 2.0.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.4.0	Flessibili

2.0.0

La tabella seguente elenca le dipendenze per la versione 2.0.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.3.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

> **Note**
>
> L'autorizzazione di sottoscrizione viene valutata durante una richiesta di sottoscrizione del client al broker MQTT locale. Se l'autorizzazione di iscrizione esistente del cliente viene revocata, il cliente non sarà più in grado di iscriversi a un argomento. Tuttavia, continuerà a ricevere messaggi da qualsiasi argomento sottoscritto in precedenza. Per evitare questo comportamento, è necessario riavviare il broker MQTT locale dopo aver revocato l'autorizzazione di sottoscrizione per forzare la riautorizzazione dei client.

Per il componente broker MQTT 5 (EMQX), aggiornate la configurazione per riavviare il broker MQTT 5. `restartIdentifier` Per ulteriori informazioni, consultate la configurazione del componente del broker MQTT 5.
Per impostazione predefinita, il componente broker MQTT 3.1.1 (Moquette) si riavvia settimanalmente quando il certificato del server cambia e i client devono effettuare una nuova autorizzazione. È possibile forzare il riavvio modificando le informazioni di connettività (indirizzi IP) del dispositivo principale o effettuando una distribuzione per rimuovere il componente broker e quindi ridistribuirlo in un secondo momento.

v2.4.5

`deviceGroups`

I gruppi di dispositivi sono gruppi di dispositivi client che dispongono delle autorizzazioni per connettersi e comunicare con un dispositivo principale. Utilizza le regole di selezione per identificare gruppi di dispositivi client e definisci politiche di autorizzazione dei dispositivi client che specificano le autorizzazioni per ciascun gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`formatVersion`

La versione del formato per questo oggetto di configurazione.

Seleziona una delle opzioni seguenti:

- `2021-03-05`

`definitions`

I gruppi di dispositivi per questo dispositivo principale. Ogni definizione specifica una regola di selezione per valutare se un dispositivo client è membro del gruppo. Ogni definizione specifica anche la politica di autorizzazione da applicare ai dispositivi client che corrispondono alla regola di selezione. Se un dispositivo client è membro di più gruppi di dispositivi, le autorizzazioni del dispositivo comprendono le politiche di autorizzazione di ciascun gruppo.

Questo oggetto contiene le seguenti informazioni:

groupNameKey

Il nome di questo gruppo di dispositivi. Sostituiscilo *groupNameKey*con un nome che ti aiuti a identificare questo gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`selectionRule`

La query che specifica quali dispositivi client sono membri di questo gruppo di dispositivi. Quando un dispositivo client si connette, il dispositivo principale valuta questa regola di selezione per determinare se il dispositivo client è membro di questo gruppo di dispositivi. Se il dispositivo client è un membro, il dispositivo principale utilizza la politica di questo gruppo di dispositivi per autorizzare le azioni del dispositivo client.

Ogni regola di selezione comprende almeno una clausola di regola di selezione, che è una singola query di espressione che può corrispondere ai dispositivi client. Le regole di selezione utilizzano la stessa sintassi di interrogazione dell'indicizzazione AWS IoT della flotta. Per ulteriori informazioni sulla sintassi delle regole di selezione, consulta la sintassi delle query di indicizzazione AWS IoT della flotta nella Guida per gli sviluppatori.AWS IoT Core

Usa il carattere * jolly per abbinare più dispositivi client con una sola clausola di regola di selezione. È possibile utilizzare questo carattere jolly all'inizio e alla fine del nome dell'oggetto per abbinare i dispositivi client i cui nomi iniziano o finiscono con la stringa specificata. Puoi anche usare questo jolly per abbinare tutti i dispositivi client.

> **Note**
>
> Per selezionare un valore che contiene i due punti (`:`), evita i due punti con una barra rovesciata (). \ In formati come JSON, è necessario evitare i caratteri della barra rovesciata, quindi è necessario immettere due caratteri di barra rovesciata prima del carattere dei due punti. Ad esempio, specificate di selezionare una `thingName: MyTeam\\:ClientDevice1` cosa il cui nome è. `MyTeam:ClientDevice1`

È possibile specificare il seguente selettore:

- `thingName`— Il nome dell' AWS IoT oggetto di un dispositivo client.

Example Esempio di regola di selezione

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi sono `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Esempio di regola di selezione (usa caratteri jolly)

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi iniziano con`MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Esempio di regola di selezione (usa caratteri jolly)

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi terminano con`MyClientDevice`.

```
thingName: *MyClientDevice
```

Example Esempio di regola di selezione (corrisponde a tutti i dispositivi)

La seguente regola di selezione corrisponde a tutti i dispositivi client.

```
thingName: *
```

`policyName`

La politica di autorizzazione che si applica ai dispositivi client di questo gruppo di dispositivi. Specificate il nome di una politica definita nell'`policies`oggetto.

`policies`

Le politiche di autorizzazione dei dispositivi client per i dispositivi client che si connettono al dispositivo principale. Ogni politica di autorizzazione specifica una serie di azioni e le risorse in cui un dispositivo client può eseguire tali azioni.

Questo oggetto contiene le seguenti informazioni:

policyNameKey

Il nome di questa politica di autorizzazione. Sostituiscilo *policyNameKey*con un nome che ti aiuti a identificare questa politica di autorizzazione. Questo nome di policy viene utilizzato per definire quale criterio si applica a un gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

statementNameKey

Il nome di questa dichiarazione politica. Sostituire *statementNameKey*con un nome che consenta di identificare questa dichiarazione politica.

Questo oggetto contiene le seguenti informazioni:

`operations`

L'elenco delle operazioni per consentire l'utilizzo delle risorse in questa politica.

È possibile includere una delle seguenti operazioni:

- `mqtt:connect`— Concede l'autorizzazione alla connessione al dispositivo principale. I dispositivi client devono disporre di questa autorizzazione per connettersi a un dispositivo principale.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:clientId:`*deviceClientId*— Limita l'accesso in base all'ID client utilizzato da un dispositivo client per connettersi al broker MQTT del dispositivo principale. Sostituisci *deviceClientId*con l'ID client da utilizzare.
- `mqtt:publish`— Concede l'autorizzazione a pubblicare messaggi MQTT sugli argomenti.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topic:`*mqttTopic*— Limita l'accesso in base all'argomento MQTT in cui un dispositivo client pubblica un messaggio. Sostituisci *MQTTTopic con l'argomento* da utilizzare.

    Questa risorsa non supporta i caratteri jolly degli argomenti MQTT.
- `mqtt:subscribe`— Concede l'autorizzazione a sottoscrivere i filtri degli argomenti MQTT per ricevere messaggi.

Questa operazione supporta le seguenti risorse:

- `mqtt:topicfilter:`*`mqttTopicFilter`*— Limita l'accesso in base agli argomenti MQTT in cui un dispositivo client può iscriversi ai messaggi. *`mqttTopicFilter`*Sostituiscilo con il filtro per argomenti da utilizzare.

  Questa risorsa supporta i caratteri jolly + degli argomenti # MQTT. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide AWS IoT Core .

  Il dispositivo client può sottoscrivere gli stessi filtri tematici consentiti. Ad esempio, se consenti al dispositivo client di sottoscrivere la `mqtt:topicfilter:client/+/status` risorsa, il dispositivo client può abbonarsi `client/+/status` ma non farlo`client/client1/status`.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le azioni.

`resources`

L'elenco delle risorse per consentire le operazioni previste da questa politica. Specificare le risorse che corrispondono alle operazioni di questa politica. Ad esempio, è possibile specificare un elenco di risorse tematiche MQTT (`mqtt:topic:`*`mqttTopic`*) in una politica che specifica l'`mqtt:publish`operazione.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le risorse. Non è possibile utilizzare il carattere * jolly per abbinare identificatori di risorse parziali. Ad esempio, puoi specificare**`"resources": "*"`**, ma non puoi specificare. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Facoltativo) Una descrizione per questa dichiarazione politica.

`certificates`

(Facoltativo) Le opzioni di configurazione del certificato per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni:

`serverCertificateValiditySeconds`

(Facoltativo) La quantità di tempo (in secondi) dopo la quale scade il certificato del server MQTT locale. È possibile configurare questa opzione per personalizzare la frequenza con cui i dispositivi client si disconnettono e si riconnettono al dispositivo principale.

Questo componente ruota il certificato del server MQTT locale 24 ore prima della scadenza. Il broker MQTT, ad esempio il componente broker MQTT Moquette, genera un nuovo certificato e si riavvia. Quando ciò accade, tutti i dispositivi client collegati a questo dispositivo principale vengono disconnessi. I dispositivi client possono riconnettersi al dispositivo principale dopo un breve periodo di tempo.

Impostazione predefinita: `604800` (7 giorni)

Valore minimo: `172800` (2 giorni)

Valore massimo: `864000` (10 giorni)

`performance`

(Facoltativo) Le opzioni di configurazione delle prestazioni per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni:

`maxActiveAuthTokens`

(Facoltativo) Il numero massimo di token di autorizzazione attivi per i dispositivi client. È possibile aumentare questo numero per consentire a un numero maggiore di dispositivi client di connettersi a un dispositivo single-core, senza riautenticarli.

Impostazione predefinita: `2500`

`cloudRequestQueueSize`

(Facoltativo) Il numero massimo di Cloud AWS richieste da mettere in coda prima che questo componente rifiuti le richieste.

Impostazione predefinita: `100`

`maxConcurrentCloudRequests`

(Facoltativo) Il numero massimo di richieste simultanee da inviare a. Cloud AWSÈ possibile aumentare questo numero per migliorare le prestazioni di autenticazione sui dispositivi principali in cui si connettono un gran numero di dispositivi client.

Impostazione predefinita: 1

`certificateAuthority`

(Facoltativo) Opzioni di configurazione dell'autorità di certificazione per sostituire l'autorità intermedia del dispositivo principale con la propria autorità di certificazione intermedia.

> **Note**
>
> Se configuri il tuo dispositivo principale Greengrass con un'autorità di certificazione (CA) personalizzata e utilizzi la stessa CA per emettere certificati per dispositivi client, Greengrass aggira i controlli delle politiche di autorizzazione per le operazioni MQTT dei dispositivi client. Il componente di autenticazione del dispositivo client si fida completamente dei client che utilizzano i certificati firmati dalla CA per cui è configurato.
> Per limitare questo comportamento quando si utilizza una CA personalizzata, crea e firma i dispositivi client utilizzando una CA o una CA intermedia diversa, quindi modifica i `certificateChainUri` campi `certificateUri` e in modo che puntino alla CA intermedia corretta.

Questo oggetto contiene le seguenti informazioni.

URI del certificato

La posizione del certificato. Può essere un URI del file system o un URI che punta a un certificato archiviato in un modulo di sicurezza hardware.

`certificateChainUri`

La posizione della catena di certificati per la CA del dispositivo principale. Questa dovrebbe essere la catena completa di certificati fino alla CA principale. Può essere un URI del file system o un URI che punta a una catena di certificati archiviata in un modulo di sicurezza hardware.

`privateKeyUri`

La posizione della chiave privata del dispositivo principale. Può essere un URI del file system o un URI che punta a una chiave privata del certificato memorizzata in un modulo di sicurezza hardware.

`security`

(Facoltativo) Opzioni di configurazione della sicurezza per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni.

`clientDeviceTrustDurationMinutes`

Durata in minuti entro la quale è possibile considerare attendibili le informazioni di autenticazione di un dispositivo client prima che sia necessario riautenticarsi con il dispositivo principale. Il valore predefinito è 1.

`metrics`

(Facoltativo) Le opzioni di metrica per questo dispositivo principale. Le metriche degli errori verranno visualizzate solo se si verifica un errore nell'autenticazione del dispositivo client. Questo oggetto contiene le seguenti informazioni:

`disableMetrics`

Se il `disableMetrics` campo è impostato su`true`, l'autenticazione del dispositivo client non raccoglierà le metriche.

Impostazione predefinita: `false`

`aggregatePeriodSeconds`

Il periodo di aggregazione in secondi che determina la frequenza con cui l'autenticazione del dispositivo client aggrega le metriche e le invia all'agente di telemetria. Ciò non modifica la frequenza con cui le metriche vengono pubblicate, poiché l'agente di telemetria le pubblica ancora una volta al giorno.

Impostazione predefinita: `3600`

startupTimeoutSeconds

(Facoltativo) Il tempo massimo in secondi di avvio del componente. Lo stato del componente cambia `BROKEN` se supera questo timeout.

Impostazione predefinita: `120`

Example Esempio: aggiornamento dell'unione della configurazione (utilizzando una politica restrittiva)

La configurazione di esempio seguente specifica di consentire ai dispositivi client i cui nomi iniziano con di connettersi e `MyClientDevice` pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
```

```
          ]
        }
      }
    }
  }
}
```

Example Esempio: configurazione merge update (utilizzando una politica permissiva)

La configurazione di esempio seguente specifica di consentire a tutti i dispositivi client di connettersi e pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

v2.4.2 - v2.4.4

`deviceGroups`

I gruppi di dispositivi sono gruppi di dispositivi client che dispongono delle autorizzazioni per connettersi e comunicare con un dispositivo principale. Utilizza le regole di selezione per

identificare gruppi di dispositivi client e definisci politiche di autorizzazione dei dispositivi client che specificano le autorizzazioni per ciascun gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`formatVersion`

La versione del formato per questo oggetto di configurazione.

Seleziona una delle opzioni seguenti:

- `2021-03-05`

`definitions`

I gruppi di dispositivi per questo dispositivo principale. Ogni definizione specifica una regola di selezione per valutare se un dispositivo client è membro del gruppo. Ogni definizione specifica anche la politica di autorizzazione da applicare ai dispositivi client che corrispondono alla regola di selezione. Se un dispositivo client è membro di più gruppi di dispositivi, le autorizzazioni del dispositivo comprendono le politiche di autorizzazione di ciascun gruppo.

Questo oggetto contiene le seguenti informazioni:

`groupNameKey`

Il nome di questo gruppo di dispositivi. Sostituiscilo *`groupNameKey`*con un nome che ti aiuti a identificare questo gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`selectionRule`

La query che specifica quali dispositivi client sono membri di questo gruppo di dispositivi. Quando un dispositivo client si connette, il dispositivo principale valuta questa regola di selezione per determinare se il dispositivo client è membro di questo gruppo di dispositivi. Se il dispositivo client è un membro, il dispositivo principale utilizza la politica di questo gruppo di dispositivi per autorizzare le azioni del dispositivo client.

Ogni regola di selezione comprende almeno una clausola di regola di selezione, che è una singola query di espressione che può corrispondere ai dispositivi client. Le regole di selezione utilizzano la stessa sintassi di interrogazione dell'indicizzazione AWS IoT della flotta. Per ulteriori informazioni sulla sintassi delle regole di selezione,

consulta la sintassi delle query di indicizzazione AWS IoT della flotta nella Guida per gli sviluppatori.AWS IoT Core

Usa il carattere * jolly per abbinare più dispositivi client con una sola clausola di regola di selezione. È possibile utilizzare questo carattere jolly alla fine del nome dell'oggetto per abbinare i dispositivi client i cui nomi iniziano con una stringa specificata dall'utente. Puoi anche usare questo jolly per abbinare tutti i dispositivi client.

> **Note**
>
> Per selezionare un valore che contiene i due punti (`:`), evita i due punti con una barra rovesciata (). `\\` In formati come JSON, è necessario evitare i caratteri della barra rovesciata, quindi è necessario immettere due caratteri di barra rovesciata prima del carattere dei due punti. Ad esempio, specificate di selezionare una `thingName: MyTeam\\\\:ClientDevice1` cosa il cui nome è. `MyTeam:ClientDevice1`

È possibile specificare il seguente selettore:

- `thingName`— Il nome dell' AWS IoT oggetto di un dispositivo client.

Example Esempio di regola di selezione

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi sono `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Esempio di regola di selezione (usa caratteri jolly)

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi iniziano con`MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Esempio di regola di selezione (corrisponde a tutti i dispositivi)

La seguente regola di selezione corrisponde a tutti i dispositivi client.

```
thingName: *
```

`policyName`

La politica di autorizzazione che si applica ai dispositivi client di questo gruppo di dispositivi. Specificate il nome di una politica definita nell'`policies`oggetto.

`policies`

Le politiche di autorizzazione dei dispositivi client per i dispositivi client che si connettono al dispositivo principale. Ogni politica di autorizzazione specifica una serie di azioni e le risorse in cui un dispositivo client può eseguire tali azioni.

Questo oggetto contiene le seguenti informazioni:

`policyNameKey`

Il nome di questa politica di autorizzazione. Sostituiscilo *`policyNameKey`*con un nome che ti aiuti a identificare questa politica di autorizzazione. Questo nome di policy viene utilizzato per definire quale criterio si applica a un gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`statementNameKey`

Il nome di questa dichiarazione politica. Sostituire *`statementNameKey`*con un nome che consenta di identificare questa dichiarazione politica.

Questo oggetto contiene le seguenti informazioni:

`operations`

L'elenco delle operazioni per consentire l'utilizzo delle risorse in questa politica.

È possibile includere una delle seguenti operazioni:

- `mqtt:connect`— Concede l'autorizzazione alla connessione al dispositivo principale. I dispositivi client devono disporre di questa autorizzazione per connettersi a un dispositivo principale.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:clientId:`*`deviceClientId`*— Limita l'accesso in base all'ID client utilizzato da un dispositivo client per connettersi al broker MQTT del dispositivo principale. Sostituisci *`deviceClientId`*con l'ID client da utilizzare.

- `mqtt:publish`— Concede l'autorizzazione a pubblicare messaggi MQTT sugli argomenti.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topic:`*`mqttTopic`*— Limita l'accesso in base all'argomento MQTT in cui un dispositivo client pubblica un messaggio. Sostituisci *`MQTTTopic con l'argomento`* da utilizzare.

    Questa risorsa non supporta i caratteri jolly degli argomenti MQTT.

- `mqtt:subscribe`— Concede l'autorizzazione a sottoscrivere i filtri degli argomenti MQTT per ricevere messaggi.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topicfilter:`*`mqttTopicFilter`*— Limita l'accesso in base agli argomenti MQTT in cui un dispositivo client può iscriversi ai messaggi. *`mqttTopicFilter`*Sostituiscilo con il filtro per argomenti da utilizzare.

    Questa risorsa supporta i caratteri jolly + degli argomenti # MQTT. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide AWS IoT Core .

    Il dispositivo client può sottoscrivere gli stessi filtri tematici consentiti. Ad esempio, se consenti al dispositivo client di sottoscrivere la `mqtt:topicfilter:client/+/status` risorsa, il dispositivo client può abbonarsi `client/+/status` ma non farlo`client/client1/status`.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le azioni.

`resources`

L'elenco delle risorse per consentire le operazioni previste da questa politica. Specificare le risorse che corrispondono alle operazioni di questa politica. Ad esempio, è possibile specificare un elenco di risorse tematiche MQTT (`mqtt:topic:`*`mqttTopic`*) in una politica che specifica l'`mqtt:publish`operazione.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le risorse. Non è possibile utilizzare il carattere * jolly per abbinare identificatori di risorse parziali. Ad esempio, puoi specificare**`"resources": "*"`**, ma non puoi specificare. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Facoltativo) Una descrizione per questa dichiarazione politica.

`certificates`

(Facoltativo) Le opzioni di configurazione del certificato per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni:

`serverCertificateValiditySeconds`

(Facoltativo) La quantità di tempo (in secondi) dopo la quale scade il certificato del server MQTT locale. È possibile configurare questa opzione per personalizzare la frequenza con cui i dispositivi client si disconnettono e si riconnettono al dispositivo principale.

Questo componente ruota il certificato del server MQTT locale 24 ore prima della scadenza. Il broker MQTT, ad esempio il componente broker MQTT Moquette, genera un nuovo certificato e si riavvia. Quando ciò accade, tutti i dispositivi client collegati a questo dispositivo principale vengono disconnessi. I dispositivi client possono riconnettersi al dispositivo principale dopo un breve periodo di tempo.

Impostazione predefinita: `604800` (7 giorni)

Valore minimo: `172800` (2 giorni)

Valore massimo: `864000` (10 giorni)

`performance`

(Facoltativo) Le opzioni di configurazione delle prestazioni per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni:

`maxActiveAuthTokens`

(Facoltativo) Il numero massimo di token di autorizzazione attivi per i dispositivi client. È possibile aumentare questo numero per consentire a un numero maggiore di dispositivi client di connettersi a un dispositivo single-core, senza riautenticarli.

Impostazione predefinita: `2500`

`cloudRequestQueueSize`

(Facoltativo) Il numero massimo di Cloud AWS richieste da mettere in coda prima che questo componente rifiuti le richieste.

Impostazione predefinita: `100`

`maxConcurrentCloudRequests`

(Facoltativo) Il numero massimo di richieste simultanee da inviare a. Cloud AWSÈ possibile aumentare questo numero per migliorare le prestazioni di autenticazione sui dispositivi principali in cui si connettono un gran numero di dispositivi client.

Impostazione predefinita: `1`

`certificateAuthority`

(Facoltativo) Opzioni di configurazione dell'autorità di certificazione per sostituire l'autorità intermedia del dispositivo principale con la propria autorità di certificazione intermedia.

> **Note**
>
> Se configuri il tuo dispositivo principale Greengrass con un'autorità di certificazione (CA) personalizzata e utilizzi la stessa CA per emettere certificati per dispositivi client, Greengrass aggira i controlli delle politiche di autorizzazione per le operazioni MQTT dei dispositivi client. Il componente di autenticazione del dispositivo client si fida completamente dei client che utilizzano i certificati firmati dalla CA per cui è configurato.
> Per limitare questo comportamento quando si utilizza una CA personalizzata, crea e firma i dispositivi client utilizzando una CA o una CA intermedia diversa, quindi modifica i `certificateChainUri` campi `certificateUri` e in modo che puntino alla CA intermedia corretta.

Questo oggetto contiene le seguenti informazioni.

URI del certificato

La posizione del certificato. Può essere un URI del file system o un URI che punta a un certificato archiviato in un modulo di sicurezza hardware.

`certificateChainUri`

La posizione della catena di certificati per la CA del dispositivo principale. Questa dovrebbe essere la catena completa di certificati fino alla CA principale. Può essere un URI del file system o un URI che punta a una catena di certificati archiviata in un modulo di sicurezza hardware.

`privateKeyUri`

La posizione della chiave privata del dispositivo principale. Può essere un URI del file system o un URI che punta a una chiave privata del certificato memorizzata in un modulo di sicurezza hardware.

`security`

(Facoltativo) Opzioni di configurazione della sicurezza per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni.

`clientDeviceTrustDurationMinutes`

Durata in minuti entro la quale è possibile considerare attendibili le informazioni di autenticazione di un dispositivo client prima che sia necessario riautenticarsi con il dispositivo principale. Il valore predefinito è 1.

`metrics`

(Facoltativo) Le opzioni di metrica per questo dispositivo principale. Le metriche degli errori verranno visualizzate solo se si verifica un errore nell'autenticazione del dispositivo client. Questo oggetto contiene le seguenti informazioni:

`disableMetrics`

Se il `disableMetrics` campo è impostato su`true`, l'autenticazione del dispositivo client non raccoglierà le metriche.

Impostazione predefinita: `false`

`aggregatePeriodSeconds`

Il periodo di aggregazione in secondi che determina la frequenza con cui l'autenticazione del dispositivo client aggrega le metriche e le invia all'agente di telemetria. Ciò non modifica la frequenza con cui le metriche vengono pubblicate, poiché l'agente di telemetria le pubblica ancora una volta al giorno.

Impostazione predefinita: `3600`

startupTimeoutSeconds

(Facoltativo) Il tempo massimo in secondi di avvio del componente. Lo stato del componente cambia `BROKEN` se supera questo timeout.

Impostazione predefinita: `120`

Example Esempio: aggiornamento dell'unione della configurazione (utilizzando una politica restrittiva)

La configurazione di esempio seguente specifica di consentire ai dispositivi client i cui nomi iniziano con di connettersi e `MyClientDevice` pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
```

```
          "resources": [
            "mqtt:topicfilter:test/topic/response"
          ]
        }
      }
    }
  }
}
```

Example Esempio: configurazione merge update (utilizzando una politica permissiva)

La configurazione di esempio seguente specifica di consentire a tutti i dispositivi client di connettersi e pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

v2.4.0 - v2.4.1

`deviceGroups`

I gruppi di dispositivi sono gruppi di dispositivi client che dispongono delle autorizzazioni per connettersi e comunicare con un dispositivo principale. Utilizza le regole di selezione per identificare gruppi di dispositivi client e definisci politiche di autorizzazione dei dispositivi client che specificano le autorizzazioni per ciascun gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`formatVersion`

La versione del formato per questo oggetto di configurazione.

Seleziona una delle opzioni seguenti:

- `2021-03-05`

`definitions`

I gruppi di dispositivi per questo dispositivo principale. Ogni definizione specifica una regola di selezione per valutare se un dispositivo client è membro del gruppo. Ogni definizione specifica anche la politica di autorizzazione da applicare ai dispositivi client che corrispondono alla regola di selezione. Se un dispositivo client è membro di più gruppi di dispositivi, le autorizzazioni del dispositivo comprendono le politiche di autorizzazione di ciascun gruppo.

Questo oggetto contiene le seguenti informazioni:

`groupNameKey`

Il nome di questo gruppo di dispositivi. Sostituiscilo *`groupNameKey`*con un nome che ti aiuti a identificare questo gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`selectionRule`

La query che specifica quali dispositivi client sono membri di questo gruppo di dispositivi. Quando un dispositivo client si connette, il dispositivo principale valuta questa regola di selezione per determinare se il dispositivo client è membro di questo gruppo di dispositivi. Se il dispositivo client è un membro, il dispositivo principale utilizza la politica di questo gruppo di dispositivi per autorizzare le azioni del dispositivo client.

Ogni regola di selezione comprende almeno una clausola di regola di selezione, che è una singola query di espressione che può corrispondere ai dispositivi client. Le regole di selezione utilizzano la stessa sintassi di interrogazione dell'indicizzazione AWS IoT della flotta. Per ulteriori informazioni sulla sintassi delle regole di selezione, consulta la sintassi delle query di indicizzazione AWS IoT della flotta nella Guida per gli sviluppatori.AWS IoT Core

Usa il carattere * jolly per abbinare più dispositivi client con una sola clausola di regola di selezione. È possibile utilizzare questo carattere jolly alla fine del nome dell'oggetto per abbinare i dispositivi client i cui nomi iniziano con una stringa specificata dall'utente. Puoi anche usare questo jolly per abbinare tutti i dispositivi client.

> **Note**
>
> Per selezionare un valore che contiene i due punti (`:`), evita i due punti con una barra rovesciata (). \\ In formati come JSON, è necessario evitare i caratteri della barra rovesciata, quindi è necessario immettere due caratteri di barra rovesciata prima del carattere dei due punti. Ad esempio, specificate di selezionare una `thingName: MyTeam\\\\:ClientDevice1` cosa il cui nome è. `MyTeam:ClientDevice1`

È possibile specificare il seguente selettore:

- `thingName`— Il nome dell' AWS IoT oggetto di un dispositivo client.

Example Esempio di regola di selezione

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi sono `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Esempio di regola di selezione (usa caratteri jolly)

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi iniziano con`MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Esempio di regola di selezione (corrisponde a tutti i dispositivi)

La seguente regola di selezione corrisponde a tutti i dispositivi client.

```
thingName: *
```

`policyName`

La politica di autorizzazione che si applica ai dispositivi client di questo gruppo di dispositivi. Specificate il nome di una politica definita nell'`policies`oggetto.

`policies`

Le politiche di autorizzazione dei dispositivi client per i dispositivi client che si connettono al dispositivo principale. Ogni politica di autorizzazione specifica una serie di azioni e le risorse in cui un dispositivo client può eseguire tali azioni.

Questo oggetto contiene le seguenti informazioni:

`policyNameKey`

Il nome di questa politica di autorizzazione. Sostituiscilo *`policyNameKey`*con un nome che ti aiuti a identificare questa politica di autorizzazione. Questo nome di policy viene utilizzato per definire quale criterio si applica a un gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`statementNameKey`

Il nome di questa dichiarazione politica. Sostituire *`statementNameKey`*con un nome che consenta di identificare questa dichiarazione politica.

Questo oggetto contiene le seguenti informazioni:

`operations`

L'elenco delle operazioni per consentire l'utilizzo delle risorse in questa politica.

È possibile includere una delle seguenti operazioni:

- `mqtt:connect`— Concede l'autorizzazione alla connessione al dispositivo principale. I dispositivi client devono disporre di questa autorizzazione per connettersi a un dispositivo principale.

  Questa operazione supporta le seguenti risorse:

- `mqtt:clientId:deviceClientId`— Limita l'accesso in base all'ID client utilizzato da un dispositivo client per connettersi al broker MQTT del dispositivo principale. Sostituisci *`deviceClientId`*con l'ID client da utilizzare.

- `mqtt:publish`— Concede l'autorizzazione a pubblicare messaggi MQTT sugli argomenti.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topic:mqttTopic`— Limita l'accesso in base all'argomento MQTT in cui un dispositivo client pubblica un messaggio. Sostituisci *`MQTTTopic con l'argomento`* da utilizzare.

    Questa risorsa non supporta i caratteri jolly degli argomenti MQTT.

- `mqtt:subscribe`— Concede l'autorizzazione a sottoscrivere i filtri degli argomenti MQTT per ricevere messaggi.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topicfilter:mqttTopicFilter`— Limita l'accesso in base agli argomenti MQTT in cui un dispositivo client può iscriversi ai messaggi. *`mqttTopicFilter`*Sostituiscilo con il filtro per argomenti da utilizzare.

    Questa risorsa supporta i caratteri jolly + degli argomenti # MQTT. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide AWS IoT Core .

    Il dispositivo client può sottoscrivere gli stessi filtri tematici consentiti. Ad esempio, se consenti al dispositivo client di sottoscrivere la `mqtt:topicfilter:client/+/status` risorsa, il dispositivo client può abbonarsi `client/+/status` ma non farlo`client/client1/status`.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le azioni.

`resources`

L'elenco delle risorse per consentire le operazioni previste da questa politica. Specificare le risorse che corrispondono alle operazioni di questa politica. Ad esempio, è possibile specificare un elenco di risorse tematiche MQTT (`mqtt:topic:mqttTopic`) in una politica che specifica l'`mqtt:publish`operazione.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le risorse. Non è possibile utilizzare il carattere * jolly per abbinare identificatori di risorse parziali. Ad esempio, puoi specificare**`"resources": "*"`**, ma non puoi specificare. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Facoltativo) Una descrizione per questa dichiarazione politica.

`certificates`

(Facoltativo) Le opzioni di configurazione del certificato per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni:

`serverCertificateValiditySeconds`

(Facoltativo) La quantità di tempo (in secondi) dopo la quale scade il certificato del server MQTT locale. È possibile configurare questa opzione per personalizzare la frequenza con cui i dispositivi client si disconnettono e si riconnettono al dispositivo principale.

Questo componente ruota il certificato del server MQTT locale 24 ore prima della scadenza. Il broker MQTT, ad esempio il componente broker MQTT Moquette, genera un nuovo certificato e si riavvia. Quando ciò accade, tutti i dispositivi client collegati a questo dispositivo principale vengono disconnessi. I dispositivi client possono riconnettersi al dispositivo principale dopo un breve periodo di tempo.

Impostazione predefinita: `604800` (7 giorni)

Valore minimo: `172800` (2 giorni)

Valore massimo: `864000` (10 giorni)

`performance`

(Facoltativo) Le opzioni di configurazione delle prestazioni per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni:

`maxActiveAuthTokens`

(Facoltativo) Il numero massimo di token di autorizzazione attivi per i dispositivi client. È possibile aumentare questo numero per consentire a un numero maggiore di dispositivi client di connettersi a un dispositivo single-core, senza riautenticarli.

Impostazione predefinita: `2500`

`cloudRequestQueueSize`

(Facoltativo) Il numero massimo di Cloud AWS richieste da mettere in coda prima che questo componente rifiuti le richieste.

Impostazione predefinita: `100`

`maxConcurrentCloudRequests`

(Facoltativo) Il numero massimo di richieste simultanee da inviare a. Cloud AWSÈ possibile aumentare questo numero per migliorare le prestazioni di autenticazione sui dispositivi principali in cui si connettono un gran numero di dispositivi client.

Impostazione predefinita: `1`

`certificateAuthority`

(Facoltativo) Opzioni di configurazione dell'autorità di certificazione per sostituire l'autorità intermedia del dispositivo principale con la propria autorità di certificazione intermedia. Questo oggetto contiene le seguenti informazioni.

Questo oggetto contiene le seguenti informazioni:

URI del certificato

La posizione del certificato. Può essere un URI del file system o un URI che punta a un certificato archiviato in un modulo di sicurezza hardware.

`certificateChainUri`

La posizione della catena di certificati per la CA del dispositivo principale. Questa dovrebbe essere la catena completa di certificati fino alla CA principale. Può essere un URI del file system o un URI che punta a una catena di certificati archiviata in un modulo di sicurezza hardware.

`privateKeyUri`

La posizione della chiave privata del dispositivo principale. Può essere un URI del file system o un URI che punta a una chiave privata del certificato memorizzata in un modulo di sicurezza hardware.

`security`

(Facoltativo) Opzioni di configurazione della sicurezza per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni.

`clientDeviceTrustDurationMinutes`

Durata in minuti entro la quale è possibile considerare attendibili le informazioni di autenticazione di un dispositivo client prima che sia necessario riautenticarsi con il dispositivo principale. Il valore predefinito è 1.

`metrics`

(Facoltativo) Le opzioni di metrica per questo dispositivo principale. Le metriche degli errori verranno visualizzate solo se si verifica un errore nell'autenticazione del dispositivo client. Questo oggetto contiene le seguenti informazioni:

`disableMetrics`

Se il `disableMetrics` campo è impostato su`true`, l'autenticazione del dispositivo client non raccoglierà le metriche.

Impostazione predefinita: `false`

`aggregatePeriodSeconds`

Il periodo di aggregazione in secondi che determina la frequenza con cui l'autenticazione del dispositivo client aggrega le metriche e le invia all'agente di telemetria. Ciò non modifica la frequenza con cui le metriche vengono pubblicate, poiché l'agente di telemetria le pubblica ancora una volta al giorno.

Impostazione predefinita: `3600`

Example Esempio: aggiornamento basato sull'unione della configurazione (utilizzando una politica restrittiva)

La configurazione di esempio seguente specifica di consentire ai dispositivi client i cui nomi iniziano con di connettersi e `MyClientDevice` pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
```

```
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
          ]
        }
      }
    }
  }
}
```

Example Esempio: configurazione merge update (utilizzando una politica permissiva)

La configurazione di esempio seguente specifica di consentire a tutti i dispositivi client di connettersi e pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
```

```
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

v2.3.x

`deviceGroups`

I gruppi di dispositivi sono gruppi di dispositivi client che dispongono delle autorizzazioni per connettersi e comunicare con un dispositivo principale. Utilizza le regole di selezione per identificare gruppi di dispositivi client e definisci politiche di autorizzazione dei dispositivi client che specificano le autorizzazioni per ciascun gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`formatVersion`

La versione del formato per questo oggetto di configurazione.

Seleziona una delle opzioni seguenti:

- `2021-03-05`

`definitions`

I gruppi di dispositivi per questo dispositivo principale. Ogni definizione specifica una regola di selezione per valutare se un dispositivo client è membro del gruppo. Ogni definizione specifica anche la politica di autorizzazione da applicare ai dispositivi client che corrispondono alla regola di selezione. Se un dispositivo client è membro di più gruppi di dispositivi, le autorizzazioni del dispositivo comprendono le politiche di autorizzazione di ciascun gruppo.

Questo oggetto contiene le seguenti informazioni:

`groupNameKey`

Il nome di questo gruppo di dispositivi. Sostituiscilo *`groupNameKey`*con un nome che ti aiuti a identificare questo gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`selectionRule`

La query che specifica quali dispositivi client sono membri di questo gruppo di dispositivi. Quando un dispositivo client si connette, il dispositivo principale valuta questa regola di selezione per determinare se il dispositivo client è membro di questo gruppo di dispositivi. Se il dispositivo client è un membro, il dispositivo principale utilizza la politica di questo gruppo di dispositivi per autorizzare le azioni del dispositivo client.

Ogni regola di selezione comprende almeno una clausola di regola di selezione, che è una singola query di espressione che può corrispondere ai dispositivi client. Le regole di selezione utilizzano la stessa sintassi di interrogazione dell'indicizzazione AWS IoT della flotta. Per ulteriori informazioni sulla sintassi delle regole di selezione, consulta la sintassi delle query di indicizzazione AWS IoT della flotta nella Guida per gli sviluppatori.AWS IoT Core

Usa il carattere * jolly per abbinare più dispositivi client con una sola clausola di regola di selezione. È possibile utilizzare questo carattere jolly alla fine del nome dell'oggetto per abbinare i dispositivi client i cui nomi iniziano con una stringa specificata dall'utente. Puoi anche usare questo jolly per abbinare tutti i dispositivi client.

> **Note**
>
> Per selezionare un valore che contiene i due punti (`:`), evita i due punti con una barra rovesciata (). \\ In formati come JSON, è necessario evitare i caratteri della barra rovesciata, quindi è necessario immettere due caratteri di barra rovesciata prima del carattere dei due punti. Ad esempio, specificate di selezionare una `thingName: MyTeam\\\\:ClientDevice1` cosa il cui nome è. `MyTeam:ClientDevice1`

È possibile specificare il seguente selettore:

- `thingName`— Il nome dell' AWS IoT oggetto di un dispositivo client.

Example Esempio di regola di selezione

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi sono `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Esempio di regola di selezione (usa caratteri jolly)

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi iniziano con`MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Esempio di regola di selezione (corrisponde a tutti i dispositivi)

La seguente regola di selezione corrisponde a tutti i dispositivi client.

```
thingName: *
```

`policyName`

La politica di autorizzazione che si applica ai dispositivi client di questo gruppo di dispositivi. Specificate il nome di una politica definita nell'`policies`oggetto.

`policies`

Le politiche di autorizzazione dei dispositivi client per i dispositivi client che si connettono al dispositivo principale. Ogni politica di autorizzazione specifica una serie di azioni e le risorse in cui un dispositivo client può eseguire tali azioni.

Questo oggetto contiene le seguenti informazioni:

`policyNameKey`

Il nome di questa politica di autorizzazione. Sostituiscilo *`policyNameKey`*con un nome che ti aiuti a identificare questa politica di autorizzazione. Questo nome di policy viene utilizzato per definire quale criterio si applica a un gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`statementNameKey`

Il nome di questa dichiarazione politica. Sostituire *`statementNameKey`*con un nome che consenta di identificare questa dichiarazione politica.

Questo oggetto contiene le seguenti informazioni:

`operations`

L'elenco delle operazioni per consentire l'utilizzo delle risorse in questa politica.

È possibile includere una delle seguenti operazioni:

- `mqtt:connect`— Concede l'autorizzazione alla connessione al dispositivo principale. I dispositivi client devono disporre di questa autorizzazione per connettersi a un dispositivo principale.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:clientId:`*`deviceClientId`*— Limita l'accesso in base all'ID client utilizzato da un dispositivo client per connettersi al broker MQTT del dispositivo principale. Sostituisci *`deviceClientId`*con l'ID client da utilizzare.
- `mqtt:publish`— Concede l'autorizzazione a pubblicare messaggi MQTT sugli argomenti.

  Questa operazione supporta le seguenti risorse:

- `mqtt:topic:`*`mqttTopic`*— Limita l'accesso in base all'argomento MQTT in cui un dispositivo client pubblica un messaggio. Sostituisci *`MQTTTopic con l'argomento`* da utilizzare.

  Questa risorsa non supporta i caratteri jolly degli argomenti MQTT.

- `mqtt:subscribe`— Concede l'autorizzazione a sottoscrivere i filtri degli argomenti MQTT per ricevere messaggi.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topicfilter:`*`mqttTopicFilter`*— Limita l'accesso in base agli argomenti MQTT in cui un dispositivo client può iscriversi ai messaggi. *`mqttTopicFilter`*Sostituiscilo con il filtro per argomenti da utilizzare.

    Questa risorsa supporta i caratteri jolly + degli argomenti # MQTT. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide AWS IoT Core .

    Il dispositivo client può sottoscrivere gli stessi filtri tematici consentiti. Ad esempio, se consenti al dispositivo client di sottoscrivere la `mqtt:topicfilter:client/+/status` risorsa, il dispositivo client può abbonarsi `client/+/status` ma non farlo`client/client1/status`.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le azioni.

`resources`

L'elenco delle risorse per consentire le operazioni previste da questa politica. Specificare le risorse che corrispondono alle operazioni di questa politica. Ad esempio, è possibile specificare un elenco di risorse tematiche MQTT (`mqtt:topic:`*`mqttTopic`*) in una politica che specifica l'`mqtt:publish`operazione.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le risorse. Non è possibile utilizzare il carattere * jolly per abbinare identificatori di risorse parziali. Ad esempio, puoi specificare**`"resources": "*"`**, ma non puoi specificare. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Facoltativo) Una descrizione per questa dichiarazione politica.

`certificates`

(Facoltativo) Le opzioni di configurazione del certificato per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni:

`serverCertificateValiditySeconds`

(Facoltativo) La quantità di tempo (in secondi) dopo la quale scade il certificato del server MQTT locale. È possibile configurare questa opzione per personalizzare la frequenza con cui i dispositivi client si disconnettono e si riconnettono al dispositivo principale.

Questo componente ruota il certificato del server MQTT locale 24 ore prima della scadenza. Il broker MQTT, ad esempio il componente broker MQTT Moquette, genera un nuovo certificato e si riavvia. Quando ciò accade, tutti i dispositivi client collegati a questo dispositivo principale vengono disconnessi. I dispositivi client possono riconnettersi al dispositivo principale dopo un breve periodo di tempo.

Impostazione predefinita: `604800` (7 giorni)

Valore minimo: `172800` (2 giorni)

Valore massimo: `864000` (10 giorni)

`performance`

(Facoltativo) Le opzioni di configurazione delle prestazioni per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni:

`maxActiveAuthTokens`

(Facoltativo) Il numero massimo di token di autorizzazione attivi per i dispositivi client. È possibile aumentare questo numero per consentire a un numero maggiore di dispositivi client di connettersi a un dispositivo single-core senza riautenticarli.

Impostazione predefinita: `2500`

`cloudRequestQueueSize`

(Facoltativo) Il numero massimo di Cloud AWS richieste da mettere in coda prima che questo componente rifiuti le richieste.

Impostazione predefinita: `100`

`maxConcurrentCloudRequests`

(Facoltativo) Il numero massimo di richieste simultanee da inviare a. Cloud AWSÈ possibile aumentare questo numero per migliorare le prestazioni di autenticazione sui dispositivi principali in cui si connettono un gran numero di dispositivi client.

Impostazione predefinita: 1

`certificateAuthority`

(Facoltativo) Opzioni di configurazione dell'autorità di certificazione per sostituire l'autorità intermedia del dispositivo principale con la propria autorità di certificazione intermedia. Questo oggetto contiene le seguenti informazioni.

URI del certificato

La posizione del certificato. Può essere un URI del file system o un URI che punta a un certificato archiviato in un modulo di sicurezza hardware.

`certificateChainUri`

La posizione della catena di certificati per la CA del dispositivo principale. Questa dovrebbe essere la catena completa di certificati fino alla CA principale. Può essere un URI del file system o un URI che punta a una catena di certificati archiviata in un modulo di sicurezza hardware.

`privateKeyUri`

La posizione della chiave privata del dispositivo principale. Può essere un URI del file system o un URI che punta a una chiave privata del certificato memorizzata in un modulo di sicurezza hardware.

`security`

(Facoltativo) Opzioni di configurazione della sicurezza per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni.

`clientDeviceTrustDurationMinutes`

Durata in minuti durante la quale è possibile considerare attendibili le informazioni di autenticazione di un dispositivo client prima che sia necessario riautenticarsi con il dispositivo principale. Il valore predefinito è 1.

Example Esempio: aggiornamento basato sull'unione della configurazione (utilizzando una politica restrittiva)

La configurazione di esempio seguente specifica di consentire ai dispositivi client i cui nomi iniziano con di connettersi e `MyClientDevice` pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
```

```
          ]
        }
      }
    }
  }
}
```

Example Esempio: configurazione merge update (utilizzando una politica permissiva)

La configurazione di esempio seguente specifica di consentire a tutti i dispositivi client di connettersi e pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

v2.2.x

`deviceGroups`

I gruppi di dispositivi sono gruppi di dispositivi client che dispongono delle autorizzazioni per connettersi e comunicare con un dispositivo principale. Utilizza le regole di selezione per

identificare gruppi di dispositivi client e definisci politiche di autorizzazione dei dispositivi client che specificano le autorizzazioni per ciascun gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`formatVersion`

La versione del formato per questo oggetto di configurazione.

Seleziona una delle opzioni seguenti:

- `2021-03-05`

`definitions`

I gruppi di dispositivi per questo dispositivo principale. Ogni definizione specifica una regola di selezione per valutare se un dispositivo client è membro del gruppo. Ogni definizione specifica anche la politica di autorizzazione da applicare ai dispositivi client che corrispondono alla regola di selezione. Se un dispositivo client è membro di più gruppi di dispositivi, le autorizzazioni del dispositivo comprendono le politiche di autorizzazione di ciascun gruppo.

Questo oggetto contiene le seguenti informazioni:

`groupNameKey`

Il nome di questo gruppo di dispositivi. Sostituiscilo *`groupNameKey`*con un nome che ti aiuti a identificare questo gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`selectionRule`

La query che specifica quali dispositivi client sono membri di questo gruppo di dispositivi. Quando un dispositivo client si connette, il dispositivo principale valuta questa regola di selezione per determinare se il dispositivo client è membro di questo gruppo di dispositivi. Se il dispositivo client è un membro, il dispositivo principale utilizza la politica di questo gruppo di dispositivi per autorizzare le azioni del dispositivo client.

Ogni regola di selezione comprende almeno una clausola di regola di selezione, che è una singola query di espressione che può corrispondere ai dispositivi client. Le regole di selezione utilizzano la stessa sintassi di interrogazione dell'indicizzazione AWS IoT della flotta. Per ulteriori informazioni sulla sintassi delle regole di selezione,

consulta la sintassi delle query di indicizzazione AWS IoT della flotta nella Guida per gli sviluppatori.AWS IoT Core

Usa il carattere * jolly per abbinare più dispositivi client con una sola clausola di regola di selezione. È possibile utilizzare questo carattere jolly alla fine del nome dell'oggetto per abbinare i dispositivi client i cui nomi iniziano con una stringa specificata dall'utente. Puoi anche usare questo jolly per abbinare tutti i dispositivi client.

> **Note**
>
> Per selezionare un valore che contiene i due punti (`:`), evita i due punti con una barra rovesciata (). \\ In formati come JSON, è necessario evitare i caratteri della barra rovesciata, quindi è necessario immettere due caratteri di barra rovesciata prima del carattere dei due punti. Ad esempio, specificate di selezionare una `thingName: MyTeam\\\\:ClientDevice1` cosa il cui nome è. `MyTeam:ClientDevice1`

È possibile specificare il seguente selettore:

- `thingName`— Il nome dell' AWS IoT oggetto di un dispositivo client.

Example Esempio di regola di selezione

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi sono `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Esempio di regola di selezione (usa caratteri jolly)

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi iniziano con`MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Esempio di regola di selezione (corrisponde a tutti i dispositivi)

La seguente regola di selezione corrisponde a tutti i dispositivi client.

```
thingName: *
```

`policyName`

La politica di autorizzazione che si applica ai dispositivi client di questo gruppo di dispositivi. Specificate il nome di una politica definita nell'`policies`oggetto.

`policies`

Le politiche di autorizzazione dei dispositivi client per i dispositivi client che si connettono al dispositivo principale. Ogni politica di autorizzazione specifica una serie di azioni e le risorse in cui un dispositivo client può eseguire tali azioni.

Questo oggetto contiene le seguenti informazioni:

`policyNameKey`

Il nome di questa politica di autorizzazione. Sostituiscilo *`policyNameKey`*con un nome che ti aiuti a identificare questa politica di autorizzazione. Questo nome di policy viene utilizzato per definire quale criterio si applica a un gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`statementNameKey`

Il nome di questa dichiarazione politica. Sostituire *`statementNameKey`*con un nome che consenta di identificare questa dichiarazione politica.

Questo oggetto contiene le seguenti informazioni:

`operations`

L'elenco delle operazioni per consentire l'utilizzo delle risorse in questa politica.

È possibile includere una delle seguenti operazioni:

- `mqtt:connect`— Concede l'autorizzazione alla connessione al dispositivo principale. I dispositivi client devono disporre di questa autorizzazione per connettersi a un dispositivo principale.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:clientId:`*`deviceClientId`*— Limita l'accesso in base all'ID client utilizzato da un dispositivo client per connettersi al broker MQTT del dispositivo principale. Sostituisci *`deviceClientId`*con l'ID client da utilizzare.

- `mqtt:publish`— Concede l'autorizzazione a pubblicare messaggi MQTT sugli argomenti.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topic:`*`mqttTopic`*— Limita l'accesso in base all'argomento MQTT in cui un dispositivo client pubblica un messaggio. Sostituisci *`MQTTTopic con l'argomento`* da utilizzare.

    Questa risorsa non supporta i caratteri jolly degli argomenti MQTT.

- `mqtt:subscribe`— Concede l'autorizzazione a sottoscrivere i filtri degli argomenti MQTT per ricevere messaggi.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topicfilter:`*`mqttTopicFilter`*— Limita l'accesso in base agli argomenti MQTT in cui un dispositivo client può iscriversi ai messaggi. *`mqttTopicFilter`*Sostituiscilo con il filtro per argomenti da utilizzare.

    Questa risorsa supporta i caratteri jolly + degli argomenti # MQTT. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide AWS IoT Core .

    Il dispositivo client può sottoscrivere gli stessi filtri tematici consentiti. Ad esempio, se consenti al dispositivo client di sottoscrivere la `mqtt:topicfilter:client/+/status` risorsa, il dispositivo client può abbonarsi `client/+/status` ma non farlo`client/client1/status`.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le azioni.

`resources`

L'elenco delle risorse per consentire le operazioni previste da questa politica. Specificare le risorse che corrispondono alle operazioni di questa politica. Ad esempio, è possibile specificare un elenco di risorse tematiche MQTT (`mqtt:topic:`*`mqttTopic`*) in una politica che specifica l'`mqtt:publish`operazione.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le risorse. Non è possibile utilizzare il carattere * jolly per abbinare identificatori di risorse parziali. Ad esempio, puoi specificare**`"resources": "*"`**, ma non puoi specificare. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Facoltativo) Una descrizione per questa dichiarazione politica.

`certificates`

(Facoltativo) Le opzioni di configurazione del certificato per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni:

`serverCertificateValiditySeconds`

(Facoltativo) La quantità di tempo (in secondi) dopo la quale scade il certificato del server MQTT locale. È possibile configurare questa opzione per personalizzare la frequenza con cui i dispositivi client si disconnettono e si riconnettono al dispositivo principale.

Questo componente ruota il certificato del server MQTT locale 24 ore prima della scadenza. Il broker MQTT, ad esempio il componente broker MQTT Moquette, genera un nuovo certificato e si riavvia. Quando ciò accade, tutti i dispositivi client collegati a questo dispositivo principale vengono disconnessi. I dispositivi client possono riconnettersi al dispositivo principale dopo un breve periodo di tempo.

Impostazione predefinita: `604800` (7 giorni)

Valore minimo: `172800` (2 giorni)

Valore massimo: `864000` (10 giorni)

`performance`

(Facoltativo) Le opzioni di configurazione delle prestazioni per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni:

`maxActiveAuthTokens`

(Facoltativo) Il numero massimo di token di autorizzazione attivi per i dispositivi client. È possibile aumentare questo numero per consentire a un numero maggiore di dispositivi client di connettersi a un dispositivo single-core senza riautenticarli.

Impostazione predefinita: `2500`

`cloudRequestQueueSize`

(Facoltativo) Il numero massimo di Cloud AWS richieste da mettere in coda prima che questo componente rifiuti le richieste.

Impostazione predefinita: `100`

`maxConcurrentCloudRequests`

(Facoltativo) Il numero massimo di richieste simultanee da inviare a. Cloud AWSÈ possibile aumentare questo numero per migliorare le prestazioni di autenticazione sui dispositivi principali in cui si connettono un gran numero di dispositivi client.

Impostazione predefinita: 1

Example Esempio: aggiornamento basato sull'unione della configurazione (utilizzando una politica restrittiva)

La configurazione di esempio seguente specifica di consentire ai dispositivi client i cui nomi iniziano con di connettersi e `MyClientDevice` pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
```

```
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
          ]
        }
      }
    }
  }
}
```

Example Esempio: configurazione merge update (utilizzando una politica permissiva)

La configurazione di esempio seguente specifica di consentire a tutti i dispositivi client di connettersi e pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
```

```
    }
  }
}
```

v2.1.x

`deviceGroups`

I gruppi di dispositivi sono gruppi di dispositivi client che dispongono delle autorizzazioni per connettersi e comunicare con un dispositivo principale. Utilizza le regole di selezione per identificare gruppi di dispositivi client e definisci politiche di autorizzazione dei dispositivi client che specificano le autorizzazioni per ciascun gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`formatVersion`

La versione del formato per questo oggetto di configurazione.

Seleziona una delle opzioni seguenti:

- `2021-03-05`

`definitions`

I gruppi di dispositivi per questo dispositivo principale. Ogni definizione specifica una regola di selezione per valutare se un dispositivo client è membro del gruppo. Ogni definizione specifica anche la politica di autorizzazione da applicare ai dispositivi client che corrispondono alla regola di selezione. Se un dispositivo client è membro di più gruppi di dispositivi, le autorizzazioni del dispositivo comprendono le politiche di autorizzazione di ciascun gruppo.

Questo oggetto contiene le seguenti informazioni:

`groupNameKey`

Il nome di questo gruppo di dispositivi. Sostituiscilo *`groupNameKey`*con un nome che ti aiuti a identificare questo gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`selectionRule`

La query che specifica quali dispositivi client sono membri di questo gruppo di dispositivi. Quando un dispositivo client si connette, il dispositivo principale valuta

questa regola di selezione per determinare se il dispositivo client è membro di questo gruppo di dispositivi. Se il dispositivo client è un membro, il dispositivo principale utilizza la politica di questo gruppo di dispositivi per autorizzare le azioni del dispositivo client.

Ogni regola di selezione comprende almeno una clausola di regola di selezione, che è una singola query di espressione che può corrispondere ai dispositivi client. Le regole di selezione utilizzano la stessa sintassi di interrogazione dell'indicizzazione AWS IoT della flotta. Per ulteriori informazioni sulla sintassi delle regole di selezione, consulta la sintassi delle query di indicizzazione AWS IoT della flotta nella Guida per gli sviluppatori.AWS IoT Core

Usa il carattere * jolly per abbinare più dispositivi client con una sola clausola di regola di selezione. È possibile utilizzare questo carattere jolly alla fine del nome dell'oggetto per abbinare i dispositivi client i cui nomi iniziano con una stringa specificata dall'utente. Puoi anche usare questo jolly per abbinare tutti i dispositivi client.

> **Note**
>
> Per selezionare un valore che contiene i due punti (`:`), evita i due punti con una barra rovesciata (). \\ In formati come JSON, è necessario evitare i caratteri della barra rovesciata, quindi è necessario immettere due caratteri di barra rovesciata prima del carattere dei due punti. Ad esempio, specificate di selezionare una `thingName: MyTeam\\\\:ClientDevice1` cosa il cui nome è. `MyTeam:ClientDevice1`

È possibile specificare il seguente selettore:

- `thingName`— Il nome dell' AWS IoT oggetto di un dispositivo client.

Example Esempio di regola di selezione

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi sono `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Esempio di regola di selezione (usa caratteri jolly)

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi iniziano con`MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Esempio di regola di selezione (corrisponde a tutti i dispositivi)

La seguente regola di selezione corrisponde a tutti i dispositivi client.

```
thingName: *
```

`policyName`

La politica di autorizzazione che si applica ai dispositivi client di questo gruppo di dispositivi. Specificate il nome di una politica definita nell'`policies`oggetto.

`policies`

Le politiche di autorizzazione dei dispositivi client per i dispositivi client che si connettono al dispositivo principale. Ogni politica di autorizzazione specifica una serie di azioni e le risorse in cui un dispositivo client può eseguire tali azioni.

Questo oggetto contiene le seguenti informazioni:

`policyNameKey`

Il nome di questa politica di autorizzazione. Sostituiscilo *`policyNameKey`*con un nome che ti aiuti a identificare questa politica di autorizzazione. Questo nome di policy viene utilizzato per definire quale criterio si applica a un gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`statementNameKey`

Il nome di questa dichiarazione politica. Sostituire *`statementNameKey`*con un nome che consenta di identificare questa dichiarazione politica.

Questo oggetto contiene le seguenti informazioni:

`operations`

L'elenco delle operazioni per consentire l'utilizzo delle risorse in questa politica.

È possibile includere una delle seguenti operazioni:

- `mqtt:connect`— Concede l'autorizzazione alla connessione al dispositivo principale. I dispositivi client devono disporre di questa autorizzazione per connettersi a un dispositivo principale.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:clientId:`*`deviceClientId`*— Limita l'accesso in base all'ID client utilizzato da un dispositivo client per connettersi al broker MQTT del dispositivo principale. Sostituisci *`deviceClientId`*con l'ID client da utilizzare.

- `mqtt:publish`— Concede l'autorizzazione a pubblicare messaggi MQTT sugli argomenti.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topic:`*`mqttTopic`*— Limita l'accesso in base all'argomento MQTT in cui un dispositivo client pubblica un messaggio. Sostituisci *`MQTTTopic con l'argomento`* da utilizzare.

    Questa risorsa non supporta i caratteri jolly degli argomenti MQTT.

- `mqtt:subscribe`— Concede l'autorizzazione a sottoscrivere i filtri degli argomenti MQTT per ricevere messaggi.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topicfilter:`*`mqttTopicFilter`*— Limita l'accesso in base agli argomenti MQTT in cui un dispositivo client può iscriversi ai messaggi. *`mqttTopicFilter`*Sostituiscilo con il filtro per argomenti da utilizzare.

    Questa risorsa supporta i caratteri jolly + degli argomenti # MQTT. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide AWS IoT Core .

    Il dispositivo client può sottoscrivere gli stessi filtri tematici consentiti. Ad esempio, se consenti al dispositivo client di sottoscrivere la `mqtt:topicfilter:client/+/status` risorsa, il dispositivo client può abbonarsi `client/+/status` ma non farlo`client/client1/status`.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le azioni.

`resources`

L'elenco delle risorse per consentire le operazioni previste da questa politica. Specificare le risorse che corrispondono alle operazioni di questa politica. Ad esempio, è possibile specificare un elenco di risorse tematiche MQTT (`mqtt:topic:`*`mqttTopic`*) in una politica che specifica l'`mqtt:publish`operazione.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le risorse. Non è possibile utilizzare il carattere * jolly per abbinare identificatori di risorse parziali. Ad esempio, puoi specificare**`"resources": "*"`**, ma non puoi specificare. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Facoltativo) Una descrizione per questa dichiarazione politica.

`certificates`

(Facoltativo) Le opzioni di configurazione del certificato per questo dispositivo principale. Questo oggetto contiene le seguenti informazioni:

`serverCertificateValiditySeconds`

(Facoltativo) La quantità di tempo (in secondi) dopo la quale scade il certificato del server MQTT locale. È possibile configurare questa opzione per personalizzare la frequenza con cui i dispositivi client si disconnettono e si riconnettono al dispositivo principale.

Questo componente ruota il certificato del server MQTT locale 24 ore prima della scadenza. Il broker MQTT, ad esempio il componente broker MQTT Moquette, genera un nuovo certificato e si riavvia. Quando ciò accade, tutti i dispositivi client collegati a questo dispositivo principale vengono disconnessi. I dispositivi client possono riconnettersi al dispositivo principale dopo un breve periodo di tempo.

Impostazione predefinita: `604800` (7 giorni)

Valore minimo: `172800` (2 giorni)

Valore massimo: `864000` (10 giorni)

Example Esempio: aggiornamento dell'unione della configurazione (utilizzando una politica restrittiva)

La configurazione di esempio seguente specifica di consentire ai dispositivi client i cui nomi iniziano con di connettersi e `MyClientDevice` pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
```

```
          ]
        }
      }
    }
  }
}
```

Example Esempio: configurazione merge update (utilizzando una politica permissiva)

La configurazione di esempio seguente specifica di consentire a tutti i dispositivi client di connettersi e pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

v2.0.x

`deviceGroups`

I gruppi di dispositivi sono gruppi di dispositivi client che dispongono delle autorizzazioni per connettersi e comunicare con un dispositivo principale. Utilizza le regole di selezione per

identificare gruppi di dispositivi client e definisci politiche di autorizzazione dei dispositivi client che specificano le autorizzazioni per ciascun gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`formatVersion`

La versione del formato per questo oggetto di configurazione.

Seleziona una delle opzioni seguenti:

- `2021-03-05`

`definitions`

I gruppi di dispositivi per questo dispositivo principale. Ogni definizione specifica una regola di selezione per valutare se un dispositivo client è membro del gruppo. Ogni definizione specifica anche la politica di autorizzazione da applicare ai dispositivi client che corrispondono alla regola di selezione. Se un dispositivo client è membro di più gruppi di dispositivi, le autorizzazioni del dispositivo comprendono le politiche di autorizzazione di ciascun gruppo.

Questo oggetto contiene le seguenti informazioni:

`groupNameKey`

Il nome di questo gruppo di dispositivi. Sostituiscilo *`groupNameKey`*con un nome che ti aiuti a identificare questo gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`selectionRule`

La query che specifica quali dispositivi client sono membri di questo gruppo di dispositivi. Quando un dispositivo client si connette, il dispositivo principale valuta questa regola di selezione per determinare se il dispositivo client è membro di questo gruppo di dispositivi. Se il dispositivo client è un membro, il dispositivo principale utilizza la politica di questo gruppo di dispositivi per autorizzare le azioni del dispositivo client.

Ogni regola di selezione comprende almeno una clausola di regola di selezione, che è una singola query di espressione che può corrispondere ai dispositivi client. Le regole di selezione utilizzano la stessa sintassi di interrogazione dell'indicizzazione AWS IoT della flotta. Per ulteriori informazioni sulla sintassi delle regole di selezione,

consulta la sintassi delle query di indicizzazione AWS IoT della flotta nella Guida per gli sviluppatori.AWS IoT Core

Usa il carattere * jolly per abbinare più dispositivi client con una sola clausola di regola di selezione. È possibile utilizzare questo carattere jolly alla fine del nome dell'oggetto per abbinare i dispositivi client i cui nomi iniziano con una stringa specificata dall'utente. Puoi anche usare questo jolly per abbinare tutti i dispositivi client.

> **Note**
>
> Per selezionare un valore che contiene i due punti (`:`), evita i due punti con una barra rovesciata (). `\\` In formati come JSON, è necessario evitare i caratteri della barra rovesciata, quindi è necessario immettere due caratteri di barra rovesciata prima del carattere dei due punti. Ad esempio, specificate di selezionare una `thingName: MyTeam\\\\:ClientDevice1` cosa il cui nome è. `MyTeam:ClientDevice1`

È possibile specificare il seguente selettore:

- `thingName`— Il nome dell' AWS IoT oggetto di un dispositivo client.

Example Esempio di regola di selezione

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi sono `MyClientDevice1` o`MyClientDevice2`.

```
thingName: MyClientDevice1 OR thingName: MyClientDevice2
```

Example Esempio di regola di selezione (usa caratteri jolly)

La seguente regola di selezione corrisponde ai dispositivi client i cui nomi iniziano con`MyClientDevice`.

```
thingName: MyClientDevice*
```

Example Esempio di regola di selezione (corrisponde a tutti i dispositivi)

La seguente regola di selezione corrisponde a tutti i dispositivi client.

```
thingName: *
```

`policyName`

La politica di autorizzazione che si applica ai dispositivi client di questo gruppo di dispositivi. Specificate il nome di una politica definita nell'`policies`oggetto.

`policies`

Le politiche di autorizzazione dei dispositivi client per i dispositivi client che si connettono al dispositivo principale. Ogni politica di autorizzazione specifica una serie di azioni e le risorse in cui un dispositivo client può eseguire tali azioni.

Questo oggetto contiene le seguenti informazioni:

`policyNameKey`

Il nome di questa politica di autorizzazione. Sostituiscilo *`policyNameKey`*con un nome che ti aiuti a identificare questa politica di autorizzazione. Questo nome di policy viene utilizzato per definire quale criterio si applica a un gruppo di dispositivi.

Questo oggetto contiene le seguenti informazioni:

`statementNameKey`

Il nome di questa dichiarazione politica. Sostituire *`statementNameKey`*con un nome che consenta di identificare questa dichiarazione politica.

Questo oggetto contiene le seguenti informazioni:

`operations`

L'elenco delle operazioni per consentire l'utilizzo delle risorse in questa politica.

È possibile includere una delle seguenti operazioni:

- `mqtt:connect`— Concede l'autorizzazione alla connessione al dispositivo principale. I dispositivi client devono disporre di questa autorizzazione per connettersi a un dispositivo principale.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:clientId:`*`deviceClientId`*— Limita l'accesso in base all'ID client utilizzato da un dispositivo client per connettersi al broker MQTT del dispositivo principale. Sostituisci *`deviceClientId`*con l'ID client da utilizzare.

- `mqtt:publish`— Concede l'autorizzazione a pubblicare messaggi MQTT sugli argomenti.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topic:`*`mqttTopic`*— Limita l'accesso in base all'argomento MQTT in cui un dispositivo client pubblica un messaggio. Sostituisci *`MQTTTopic con l'argomento`* da utilizzare.

    Questa risorsa non supporta i caratteri jolly degli argomenti MQTT.

- `mqtt:subscribe`— Concede l'autorizzazione a sottoscrivere i filtri degli argomenti MQTT per ricevere messaggi.

  Questa operazione supporta le seguenti risorse:

  - `mqtt:topicfilter:`*`mqttTopicFilter`*— Limita l'accesso in base agli argomenti MQTT in cui un dispositivo client può iscriversi ai messaggi. *`mqttTopicFilter`*Sostituiscilo con il filtro per argomenti da utilizzare.

    Questa risorsa supporta i caratteri jolly + degli argomenti # MQTT. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide AWS IoT Core .

    Il dispositivo client può sottoscrivere gli stessi filtri tematici consentiti. Ad esempio, se consenti al dispositivo client di sottoscrivere la `mqtt:topicfilter:client/+/status` risorsa, il dispositivo client può abbonarsi `client/+/status` ma non farlo`client/client1/status`.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le azioni.

`resources`

L'elenco delle risorse per consentire le operazioni previste da questa politica. Specificare le risorse che corrispondono alle operazioni di questa politica. Ad esempio, è possibile specificare un elenco di risorse tematiche MQTT (`mqtt:topic:`*`mqttTopic`*) in una politica che specifica l'`mqtt:publish`operazione.

È possibile specificare il carattere * jolly per consentire l'accesso a tutte le risorse. Non è possibile utilizzare il carattere * jolly per abbinare identificatori di risorse parziali. Ad esempio, puoi specificare**`"resources": "*"`**, ma non puoi specificare. **`"resources": "mqtt:clientId:*"`**

`statementDescription`

(Facoltativo) Una descrizione per questa dichiarazione politica.

Example Esempio: aggiornamento dell'unione della configurazione (utilizzando una politica restrittiva)

La configurazione di esempio seguente specifica di consentire ai dispositivi client i cui nomi iniziano con di connettersi e `MyClientDevice` pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyDeviceGroup": {
        "selectionRule": "thingName: MyClientDevice*",
        "policyName": "MyRestrictivePolicy"
      }
    },
    "policies": {
      "MyRestrictivePolicy": {
        "AllowConnect": {
          "statementDescription": "Allow client devices to connect.",
          "operations": [
            "mqtt:connect"
          ],
          "resources": [
            "*"
          ]
        },
        "AllowPublish": {
          "statementDescription": "Allow client devices to publish on test/topic.",
          "operations": [
            "mqtt:publish"
          ],
          "resources": [
            "mqtt:topic:test/topic"
          ]
        },
        "AllowSubscribe": {
          "statementDescription": "Allow client devices to subscribe to test/topic/
response.",
          "operations": [
```

```
            "mqtt:subscribe"
          ],
          "resources": [
            "mqtt:topicfilter:test/topic/response"
          ]
        }
      }
    }
  }
}
```

Example Esempio: configurazione merge update (utilizzando una politica permissiva)

La configurazione di esempio seguente specifica di consentire a tutti i dispositivi client di connettersi e pubblicare/sottoscrivere su tutti gli argomenti.

```
{
  "deviceGroups": {
    "formatVersion": "2021-03-05",
    "definitions": {
      "MyPermissiveDeviceGroup": {
        "selectionRule": "thingName: *",
        "policyName": "MyPermissivePolicy"
      }
    },
    "policies": {
      "MyPermissivePolicy": {
        "AllowAll": {
          "statementDescription": "Allow client devices to perform all actions.",
          "operations": [
            "*"
          ],
          "resources": [
            "*"
          ]
        }
      }
    }
  }
}
```

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.4.5	Nuove funzionalità Aggiunge il supporto per i prefissi jolly per la selezione dei nomi degli oggetti con il parametro. `selectionRule`

Versione	Modifiche
	Correzioni di bug e miglioramenti Risolve un problema per cui i certificati non vengono aggiornati con nuove informazioni di connettività in alcuni casi.
2.4.4	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.4.3	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.4.2	Nuove funzionalità Aggiunge una nuova opzione di configurazione. `startupTimeoutSeconds`
2.4.1	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.4.0	Nuove funzionalità • Aggiunge il supporto per l'autenticazione dei dispositivi client per l'emissione di metriche operative che verranno pubblicate dall'agente di telemetria. Correzioni di bug e miglioramenti • Risolve un problema per cui l'autenticazione del dispositivo client impiega più di 10 secondi per verificare l'identità del dispositivo client. • Correzioni e miglioramenti minori aggiuntivi.
2.3.2	Correzioni di bug e miglioramenti • Aggiunge il supporto per la memorizzazione nella cache delle informazioni sul nome host in modo che il componente generi correttamente gli oggetti del certificato quando viene riavviato in modalità offline.
2.3.1	Correzioni di bug e miglioramenti • Risolve una perdita di memoria.

Versione	Modifiche
2.3.0	**Warning** Questa versione non è più disponibile. I miglioramenti di questa versione sono disponibili nelle versioni successive di questo componente. Nuove funzionalità • Aggiunge il supporto per l'autenticazione offline dei dispositivi client in modo che possano continuare a connettersi al dispositivo principale quando il dispositivo principale non è connesso a Internet. • Aggiunge il supporto per l'autorità di certificazione fornita dal cliente che il dispositivo principale utilizza come certificato principale per generare certificati broker MQTT.
2.2.3	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.2.2	Correzioni di bug e miglioramenti • Risolve un problema per cui il certificato del server MQTT locale ruota più spesso del previsto in determinati scenari.
2.2.1	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.2.0	Nuove funzionalità • Aggiunge il supporto per componenti personalizzati per chiamare le operazioni di comunicazione tra processi (IPC) per autenticare e autorizzare i dispositivi client. È possibile utilizzare queste operazioni in un componente broker MQTT personalizzato, ad esempio. Per ulteriori informazioni, consulta IPC: autenticazione e autorizzazione dei dispositivi client. • Aggiunge le `threadPoolSize` opzioni `maxActiveAuthTokens` `cloudQueueSize` , e che è possibile configurare per ottimizzare le prestazioni di questo componente.

Versione	Modifiche
2.1.0	Nuove funzionalità • Aggiunge l'`serverCertificateValiditySeconds` opzione che è possibile configurare per personalizzare alla scadenza del certifica to del server del broker MQTT. È possibile configurare la scadenza del certificato del server dopo 2-10 giorni. Correzioni di bug e miglioramenti • Risolve i problemi relativi al modo in cui questo componente gestisce gli aggiornamenti di ripristino della configurazione. • Risolve un problema per cui il certificato del server MQTT locale ruota più spesso del previsto in determinati scenari. Per applicare questa correzione, è necessario utilizzare anche la versione 2.1.0 o successiva del componente broker Moquette MQTT. • Migliora i messaggi che questo componente registra quando ruota i certificati. • Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.0.4	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.0.3	Correzioni di bug e miglioramenti • Le credenziali ora si aggiornano se si ruota la chiave privata del dispositi vo principale. • Aggiornamenti per rendere più chiari i messaggi di registro.
2.0.2	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.1	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.0.0	Versione iniziale.

CloudWatch metriche

Il componente Amazon CloudWatch metrics (`aws.greengrass.Cloudwatch`) pubblica metriche personalizzate dai dispositivi core Greengrass su Amazon. CloudWatch Il componente consente

ai componenti di pubblicare CloudWatch metriche, che possono essere utilizzate per monitorare e analizzare l'ambiente del dispositivo principale Greengrass. Per ulteriori informazioni, consulta Using Amazon CloudWatch metrics nella Amazon CloudWatch User Guide.

Per pubblicare una CloudWatch metrica con questo componente, pubblica un messaggio su un argomento a cui questo componente è abbonato. Per impostazione predefinita, questo componente sottoscrive l'argomento di pubblicazione/sottoscrizione `cloudwatch/metric/put` locale. È possibile specificare altri argomenti, inclusi gli argomenti AWS IoT Core MQTT, quando si distribuisce questo componente.

Questo componente raggruppa in batch le metriche che si trovano nello stesso spazio dei nomi e le pubblica a intervalli regolari. CloudWatch

> **Note**
>
> Questo componente offre funzionalità simili al connettore Metrics in V1. CloudWatch AWS IoT Greengrass Per ulteriori informazioni, consulta CloudWatch Metrics Connector nella AWS IoT Greengrass V1 Developer Guide.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 3.1.x
- 3.0.x
- 2.1.x
- 2,0x

Per informazioni sulle modifiche apportate a ciascuna versione del componente, consulta il changelog.

Type

v3.x

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

v2.x

Questo componente è un componente Lambda () `aws.greengrass.lambda`. Il nucleo Greengrass esegue la funzione Lambda di questo componente utilizzando il componente di avvio Lambda.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

v3.x

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

v2.x

Questo componente può essere installato solo sui dispositivi principali Linux.

Requisiti

Questo componente ha i seguenti requisiti:

3.x

- Python versione 3.7 installata sul dispositivo principale e aggiunta alla variabile di ambiente PATH.
- Il ruolo del dispositivo Greengrass deve consentire l'`cloudwatch:PutMetricData`azione, come illustrato nel seguente esempio di policy IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Per ulteriori informazioni, consulta il riferimento CloudWatch alle autorizzazioni di Amazon nella Amazon CloudWatch User Guide.

2.x

- Il dispositivo principale deve soddisfare i requisiti per eseguire le funzioni Lambda. Se desideri che il dispositivo principale esegua funzioni Lambda containerizzate, il dispositivo deve soddisfare i requisiti per farlo. Per ulteriori informazioni, consulta Requisiti della funzione Lambda.
- Python versione 3.7 installata sul dispositivo principale e aggiunta alla variabile di ambiente PATH.
- Il ruolo del dispositivo Greengrass deve consentire l'`cloudwatch:PutMetricData`azione, come illustrato nel seguente esempio di policy IAM.

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Per ulteriori informazioni, consulta il riferimento CloudWatch alle autorizzazioni di Amazon nella Amazon CloudWatch User Guide.

- Per ricevere i dati di output da questo componente, devi unire il seguente aggiornamento di configurazione per il componente legacy del router di abbonamento (`aws.greengrass.LegacySubscriptionRouter`) quando distribuisci questo componente. Questa configurazione specifica l'argomento in cui questo componente pubblica le risposte.

  Legacy subscription router v2.1.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-cloudwatch": {
        "id": "aws-greengrass-cloudwatch",
        "source": "component:aws.greengrass.Cloudwatch",
        "subject": "cloudwatch/metric/put/status",
        "target": "cloud"
      }
    }
  }
  ```

  Legacy subscription router v2.0.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-cloudwatch": {
        "id": "aws-greengrass-cloudwatch",
        "source": "arn:aws:lambda:region:aws:function:aws-greengrass-
  cloudwatch:version",
        "subject": "cloudwatch/metric/put/status",
        "target": "cloud"
  ```

```
    }
  }
}
```

- Sostituisci la *regione* con quella Regione AWS che usi.
- Sostituisci la *versione* con la versione della funzione Lambda eseguita da questo componente. Per trovare la versione della funzione Lambda, è necessario visualizzare la ricetta per la versione di questo componente che si desidera distribuire. Apri la pagina dei dettagli di questo componente nella AWS IoT Greengrass console e cerca la coppia chiave-valore della funzione Lambda. Questa coppia chiave-valore contiene il nome e la versione della funzione Lambda.

> **Important**
>
> È necessario aggiornare la versione della funzione Lambda sul router di abbonamento legacy ogni volta che si distribuisce questo componente. Ciò garantisce l'utilizzo della versione corretta della funzione Lambda per la versione del componente che si distribuisce.

Per ulteriori informazioni, consulta Creare distribuzione.

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`monitoring.` *region*`.amazonaws.com`	443	Sì	Metriche di caricamento. CloudWatch

Dipendenze

Quando distribuisci un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle sue dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

3.0.0 - 3.1.0

La tabella seguente elenca le dipendenze per le versioni da 3.0.0 a 3.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <3.0.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.1.2 and 2.1.3

La tabella seguente elenca le dipendenze per le versioni 2.1.2 e 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.8 - 2.1.0

La tabella seguente elenca le dipendenze per le versioni da 2.0.8 a 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.7

La tabella seguente elenca le dipendenze per la versione 2.0.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.6

La tabella seguente elenca le dipendenze per la versione 2.0.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.5

La tabella seguente elenca le dipendenze per la versione 2.0.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.4

La tabella seguente elenca le dipendenze per la versione 2.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili

Dipendenza	Versioni compatibili	Tipo di dipendenza
Servizio di scambio di token	^2.0.0	Rigidi

2.0.3

La tabella seguente elenca le dipendenze per la versione 2.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Rigidi
Lanciatore Lambda	>=1.0.0	Rigidi
Runtime Lambda	>=1.0.0	Flessibili
Servizio di scambio di token	>=1.0.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

v3.x

`PublishInterval`

(Facoltativo) Il numero massimo di secondi di attesa prima che il componente pubblichi le metriche in batch per un determinato namespace. Per configurare il componente in modo che pubblichi le metriche man mano che le riceve, ovvero senza raggruppamento in batch, specificate. `0`

Il componente viene pubblicato CloudWatch dopo aver ricevuto 20 metriche nello stesso spazio dei nomi o dopo l'intervallo specificato.

> **Note**
>
> Il componente non specifica l'ordine di pubblicazione degli eventi.

Questo valore può essere al massimo di 900 secondi.

Impostazione predefinita: 10 secondi

`MaxMetricsToRetain`

(Facoltativo) Il numero massimo di metriche in tutti i namespace da salvare in memoria prima che il componente le sostituisca con metriche più recenti.

Questo limite si applica quando il dispositivo principale non dispone di una connessione a Internet, quindi il componente memorizza nel buffer le metriche per pubblicarle in un secondo momento. Quando il buffer è pieno, il componente sostituisce le metriche più vecchie con quelle più recenti. Le metriche in un determinato spazio dei nomi sostituiscono solo le metriche nello stesso spazio dei nomi.

> **Note**
>
> Se il processo host per il componente viene interrotto, il componente non salva le metriche. Ciò può accadere durante una distribuzione o al riavvio del dispositivo principale, ad esempio.

Questo valore deve essere almeno 2.000 parametri.

Impostazione predefinita: 5.000 metriche

`InputTopic`

(Facoltativo) L'argomento a cui il componente si iscrive per ricevere messaggi. Se si specifica `true` for`PubSubToIoTCore`, è possibile utilizzare i caratteri jolly MQTT (+ e #) in questo argomento.

Impostazione predefinita: `cloudwatch/metric/put`

`OutputTopic`

(Facoltativo) L'argomento su cui il componente pubblica le risposte sullo stato.

Impostazione predefinita: `cloudwatch/metric/put/status`

`PubSubToIoTCore`

(Facoltativo) Valore di stringa che definisce se pubblicare e sottoscrivere argomenti AWS IoT Core MQTT. I valori supportati sono `true` e `false`.

Impostazione predefinita: `false`

`UseInstaller`

(Facoltativo) Valore booleano che definisce se utilizzare lo script di installazione in questo componente per installare le dipendenze SDK di questo componente.

Imposta questo valore su `false` se desideri utilizzare uno script personalizzato per installare le dipendenze o se desideri includere le dipendenze di runtime in un'immagine Linux predefinita. Per utilizzare questo componente, è necessario installare le seguenti librerie, comprese le eventuali dipendenze, e renderle disponibili all'utente predefinito del sistema Greengrass.

- SDK per dispositivi AWS IoT v2 per Python
- AWS SDK for Python (Boto3)

Impostazione predefinita: `true`

`PublishRegion`

(Facoltativo) L'oggetto Regione AWS su cui pubblicare CloudWatch le metriche. Questo valore sostituisce la regione predefinita per il dispositivo principale. Questo parametro è obbligatorio solo per le metriche interregionali.

`accessControl`

(Facoltativo) L'oggetto che contiene la politica di autorizzazione che consente al componente di pubblicare e sottoscrivere gli argomenti specificati. Se si specificano valori personalizzati per `InputTopic` e`OutputTopic`, è necessario aggiornare i valori delle risorse in questo oggetto.

Impostazione predefinita:

```
{
  "aws.greengrass.ipc.pubsub": {
    "aws.greengrass.Cloudwatch:pubsub:1": {
      "policyDescription": "Allows access to subscribe to input topics.",
      "operations": [
```

```
        "aws.greengrass#SubscribeToTopic"
      ],
      "resources": [
        "cloudwatch/metric/put"
      ]
    },
    "aws.greengrass.Cloudwatch:pubsub:2": {
      "policyDescription": "Allows access to publish to output topics.",
      "operations": [
        "aws.greengrass#PublishToTopic"
      ],
      "resources": [
        "cloudwatch/metric/put/status"
      ]
    }
  },
  "aws.greengrass.ipc.mqttproxy": {
    "aws.greengrass.Cloudwatch:mqttproxy:1": {
      "policyDescription": "Allows access to subscribe to input topics.",
      "operations": [
        "aws.greengrass#SubscribeToIoTCore"
      ],
      "resources": [
        "cloudwatch/metric/put"
      ]
    },
    "aws.greengrass.Cloudwatch:mqttproxy:2": {
      "policyDescription": "Allows access to publish to output topics.",
      "operations": [
        "aws.greengrass#PublishToIoTCore"
      ],
      "resources": [
        "cloudwatch/metric/put/status"
      ]
    }
  }
}
```

Example Esempio: fusione e aggiornamento della configurazione

```
{
  "PublishInterval": 0,
```

```
  "PubSubToIoTCore": true
}
```

v2.x

> **Note**
>
> La configurazione predefinita di questo componente include i parametri della funzione Lambda. Ti consigliamo di modificare solo i seguenti parametri per configurare questo componente sui tuoi dispositivi.

`lambdaParams`

Un oggetto che contiene i parametri per la funzione Lambda di questo componente. Questo oggetto contiene le seguenti informazioni:

`EnvironmentVariables`

Un oggetto che contiene i parametri della funzione Lambda. Questo oggetto contiene le seguenti informazioni:

`PUBLISH_INTERVAL`

(Facoltativo) Il numero massimo di secondi di attesa prima che il componente pubblichi le metriche in batch per un determinato namespace. Per configurare il componente in modo che pubblichi le metriche man mano che le riceve, ovvero senza raggruppamento in batch, specificate. `0`

Il componente viene pubblicato CloudWatch dopo aver ricevuto 20 metriche nello stesso spazio dei nomi o dopo l'intervallo specificato.

> **Note**
>
> Il componente non garantisce l'ordine di pubblicazione degli eventi.

Questo valore può essere al massimo di 900 secondi.

Impostazione predefinita: 10 secondi

`MAX_METRICS_TO_RETAIN`

(Facoltativo) Il numero massimo di metriche in tutti i namespace da salvare in memoria prima che il componente le sostituisca con metriche più recenti.

Questo limite si applica quando il dispositivo principale non dispone di una connessione a Internet, quindi il componente memorizza nel buffer le metriche per pubblicarle in un secondo momento. Quando il buffer è pieno, il componente sostituisce le metriche più vecchie con quelle più recenti. Le metriche in un determinato spazio dei nomi sostituiscono solo le metriche nello stesso spazio dei nomi.

> **Note**
>
> Se il processo host per il componente viene interrotto, il componente non salva le metriche. Ciò può accadere durante una distribuzione o al riavvio del dispositivo principale, ad esempio.

Questo valore deve essere almeno 2.000 parametri.

Impostazione predefinita: 5.000 metriche

`PUBLISH_REGION`

(Facoltativo) L'oggetto Regione AWS su cui pubblicare le CloudWatch metriche. Questo valore sostituisce la regione predefinita per il dispositivo principale. Questo parametro è obbligatorio solo per le metriche interregionali.

`containerMode`

(Facoltativo) La modalità di containerizzazione per questo componente. Seleziona una delle opzioni seguenti:

- `NoContainer`— Il componente non viene eseguito in un ambiente di runtime isolato.
- `GreengrassContainer`— Il componente viene eseguito in un ambiente di runtime isolato all'interno del AWS IoT Greengrass contenitore.

Impostazione predefinita: `GreengrassContainer`

`containerParams`

(Facoltativo) Un oggetto che contiene i parametri del contenitore per questo componente. Il componente utilizza questi parametri se si specifica `GreengrassContainer` per`containerMode`.

Questo oggetto contiene le seguenti informazioni:

`memorySize`

(Facoltativo) La quantità di memoria (in kilobyte) da allocare al componente.

Il valore predefinito è 64 MB (65.535 KB).

`pubsubTopics`

(Facoltativo) Un oggetto che contiene gli argomenti a cui il componente sottoscrive la sottoscrizione per ricevere messaggi. È possibile specificare ogni argomento e se il componente sottoscrive gli argomenti MQTT AWS IoT Core o gli argomenti di pubblicazione/sottoscrizione locali.

Questo oggetto contiene le seguenti informazioni:

`0`— Si tratta di un indice di matrice sotto forma di stringa.

Un oggetto che contiene le seguenti informazioni:

`type`

(Facoltativo) Il tipo di messaggi di pubblicazione/sottoscrizione utilizzato da questo componente per sottoscrivere i messaggi. Seleziona una delle opzioni seguenti:

- `PUB_SUB`: iscriviti ai messaggi di pubblicazione/sottoscrizione locali. Se scegli questa opzione, l'argomento non può contenere caratteri jolly MQTT. Per ulteriori informazioni su come inviare messaggi dal componente personalizzato quando si specifica questa opzione, vedere. Pubblicare/sottoscrivere messaggi locali
- `IOT_CORE`— Abbonarsi ai messaggi AWS IoT Core MQTT. Se scegli questa opzione, l'argomento può contenere caratteri jolly MQTT. Per ulteriori informazioni su come inviare messaggi da componenti personalizzati quando si specifica questa opzione, vedere. AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

Impostazione predefinita: `PUB_SUB`

`topic`

(Facoltativo) L'argomento a cui il componente si iscrive per ricevere messaggi. Se si specifica `IotCore` for`type`, è possibile utilizzare i caratteri jolly MQTT (+and#) in questo argomento.

Example Esempio: aggiornamento basato sull'unione della configurazione (modalità contenitore)

```
{
  "containerMode": "GreengrassContainer"
}
```

Example Esempio: aggiornamento tramite fusione della configurazione (nessuna modalità contenitore)

```
{
  "containerMode": "NoContainer"
}
```

Dati di input

Questo componente accetta le metriche sul seguente argomento e le pubblica su. CloudWatch Per impostazione predefinita, questo componente sottoscrive i messaggi di pubblicazione/sottoscrizione locali. Per ulteriori informazioni su come pubblicare messaggi su questo componente dai componenti personalizzati, vedere. Pubblicare/sottoscrivere messaggi locali

A partire dalla versione del componente v3.0.0, è possibile facoltativamente configurare questo componente per sottoscrivere un argomento MQTT impostando il parametro di configurazione su. `PubSubToIoTCore` `true` Per ulteriori informazioni sulla pubblicazione di messaggi su un argomento MQTT nei componenti personalizzati, vedere. AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

Argomento predefinito: `cloudwatch/metric/put`

Il messaggio accetta le seguenti proprietà. I messaggi di input devono essere in formato JSON.

`request`

La metrica in questo messaggio.

L'oggetto della richiesta contiene i dati dei parametri da pubblicare in CloudWatch. I valori metrici devono soddisfare le specifiche dell'`PutMetricData`operazione.

Tipo: `object` che contiene le seguenti informazioni:

`namespace`

Lo spazio dei nomi definito dall'utente per i dati metrici in questa richiesta. CloudWatch utilizza i namespace come contenitori per i punti dati metrici.

> **Note**
>
> Non è possibile specificare uno spazio dei nomi che inizi con la stringa riservata. `AWS/`

Tipo: `string`

Modello valido: `[^:].*`

`metricData`

I dati del parametro.

Tipo: `object` che contiene le seguenti informazioni:

`metricName`

Nome del parametro.

Tipo: `string`

`value`

Il valore del parametro.

> **Note**
>
> CloudWatch rifiuta valori troppo piccoli o troppo grandi. Il valore deve essere compreso tra `8.515920e-109` e `1.174271e+108` (Base 10) o `2e-360` e `2e360` (Base 2). CloudWatch non supporta valori speciali come `NaN`+`Infinity`, e-`Infinity`.

Tipo: `double`

`dimensions`

(Facoltativo) Le dimensioni della metrica. Le dimensioni forniscono ulteriori informazioni sul parametro e sui relativi dati. Un parametro è in grado di definire fino a 10 dimensioni.

Questo componente include automaticamente una dimensione denominata`coreName`, dove il valore è il nome del dispositivo principale.

Tipo: `array` di oggetti che contengono ciascuno le seguenti informazioni:

`name`

(Facoltativo) Il nome della dimensione.

Tipo: `string`

`value`

(Facoltativo) Il valore della dimensione.

Tipo: `string`

`timestamp`

(Facoltativo) L'ora in cui sono stati ricevuti i dati metrici, espressa in secondi nell'epoca Unix.

Il valore predefinito è l'ora in cui il componente riceve il messaggio.

Tipo: `double`

> **Note**
>
> Se utilizzi tra le versioni 2.0.3 e 2.0.7 di questo componente, ti consigliamo di recuperare il timestamp separatamente per ogni metrica quando invii più metriche da un'unica fonte. Non utilizzare una variabile per memorizzare il timestamp.

`unit`

(Facoltativo) L'unità della metrica.

Tipo: `string`

Valori validi:
`Seconds`,`Microseconds`,`Milliseconds`,`Bytes`,`Kilobytes`,`Megabytes`,`Gigabytes`,`Terabyt`
`Second`,`Kilobytes/Second`,`Megabytes/Second`,`Gigabytes/Second`,`Terabytes/`
`Second`,`Bits/Second`,`Kilobits/Second`,`Megabits/Second`,`Gigabits/`
`Second`,`Terabits/Second`, `Count/Second` `None`

L'impostazione predefinita è `None`.

> **Note**
>
> Tutte le quote che si applicano all' CloudWatch `PutMetricData`API si applicano alle metriche pubblicate con questo componente. Le seguenti quote sono particolarmente importanti:
>
> - Limite di 40 KB per il payload dell'API
> - 20 parametri per ciascuna richiesta API
> - 150 transazioni al secondo (TPS) per l'API `PutMetricData`
>
> Per ulteriori informazioni, consulta le quote CloudWatch di servizio nella Guida per l'CloudWatch utente.

Example Input di esempio

```
{
  "request": {
    "namespace": "Greengrass",
    "metricData": {
      "metricName": "latency",
      "dimensions": [
        {
          "name": "hostname",
          "value": "test_hostname"
        }
      ],
      "timestamp": 1539027324,
      "value": 123.0,
      "unit": "Seconds"
    }
```

```
  }
}
```

Dati di output

Per impostazione predefinita, questo componente pubblica le risposte come dati di output sul seguente argomento di pubblicazione/sottoscrizione locale. Per ulteriori informazioni su come sottoscrivere i messaggi relativi a questo argomento nei componenti personalizzati, consulta. Pubblicare/sottoscrivere messaggi locali

Facoltativamente, è possibile configurare questo componente per la pubblicazione su un argomento MQTT impostando il parametro di `PubSubToIoTCore` configurazione su. `true` Per ulteriori informazioni sulla sottoscrizione ai messaggi su un argomento MQTT nei componenti personalizzati, vedere. AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

> **Note**
>
> Per impostazione predefinita, le versioni dei componenti 2.0.x pubblicano le risposte come dati di output su un argomento MQTT. È necessario specificare l'argomento come contenuto `subject` nella configurazione del componente legacy del router di abbonamento.

Argomento predefinito: `cloudwatch/metric/put/status`

Example Output di esempio: Operazione riuscita

La risposta include lo spazio dei nomi dei dati metrici e il `RequestId` campo della risposta. CloudWatch

```
{
  "response": {
    "cloudwatch_rid": "70573243-d723-11e8-b095-75ff2EXAMPLE",
    "namespace": "Greengrass",
    "status": "success"
  }
}
```

Example Esempio di output: Errore

```
{
```

```
  "response" : {
    "namespace": "Greengrass",
    "error": "InvalidInputException",
    "error_message": "cw metric is invalid",
    "status": "fail"
  }
}
```

> **Note**
>
> Se il componente rileva un errore che può essere riprovato, ad esempio un errore di connessione, riprova la pubblicazione nel batch successivo.

Licenze

Questo componente include i seguenti software/licenze di terze parti:

- AWS SDK for Python (Boto3)/Apache License 2.0
- botocore/Apache License 2.0
- dateutil/PSF License
- docutils/BSD License, GNU General Public License (GPL), Python Software Foundation License, Public Domain
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/MIT License

Questo componente è rilasciato in base al contratto di licenza del software Greengrass Core.

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.Cloudwatch.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.Cloudwatch.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.greengrass.Cloudwatch.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.Cloudwatch.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

v3.x

Versione	Modifiche
3.1.0	Correzioni di bug e miglioramenti • Aggiunge il supporto per le configurazioni proxy di rete HTTPS. Per ulteriori informazioni, consultare Connessione alla porta 443 o tramite un proxy di rete e Abilita il dispositivo principale in modo che consideri attendibile un proxy HTTPS.
3.0.0	Questa versione del componente CloudWatch metrics prevede parametri di configurazione diversi rispetto alla versione 2.x. Se utilizzi una configura zione non predefinita per la versione 2.x e desideri eseguire l'aggiorn amento dalla v2.x alla v3.x, devi aggiornare la configurazione del

Versione	Modifiche
	componente. Per ulteriori informazioni, consulta Configurazione del componente metrics. CloudWatch Nuove funzionalità • Aggiunge il supporto per i dispositivi principali che eseguono Windows. • Cambia il tipo di componente da componente Lambda a component e generico. Questo componente ora non dipende più dal componente legacy del router di sottoscrizione per la creazione di abbonamenti. • Aggiunge un nuovo parametro di `InputTopic` configurazione per specificare l'argomento a cui il componente si iscrive per ricevere messaggi. • Aggiunge un nuovo parametro di `OutputTopic` configurazione per specificare l'argomento in cui il componente pubblica le risposte di stato. • Aggiunge un nuovo parametro di `PubSubToIoTCore` configura zione per specificare se pubblicare e sottoscrivere argomenti AWS IoT Core MQTT. • Aggiunge il nuovo parametro `UseInstaller` di configurazione che consente di disabilitare facoltativamente lo script di installazione che installa le dipendenze dei componenti. Correzioni di bug e miglioramenti Aggiunge il supporto per timestamp duplicati nei dati di input.

v2.x

Versione	Modifiche
2.1.3	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.1	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.

Versione	Modifiche
2.1.0	Nuove funzionalità • Aggiunge il supporto per le configurazioni proxy di rete HTTPS. Per ulteriori informazioni, consultare Connessione alla porta 443 o tramite un proxy di rete e Abilita il dispositivo principale in modo che consideri attendibile un proxy HTTPS.
2.0.8	Correzioni di bug e miglioramenti • Aggiunge il supporto per timestamp duplicati nei dati di input. • Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.0.7	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.6	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.0.5	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.0.4	Versione aggiornata per la versione 2.1.0 di Greengrass nucleus.
2.0.3	Versione iniziale.

Consulta anche

- Utilizzo dei CloudWatch parametri di Amazon nella Amazon CloudWatch User Guide
- PutMetricDatanell'Amazon CloudWatch API Reference

AWS IoT Device Defender

Il AWS IoT Device Defender componente (`aws.greengrass.DeviceDefender`) notifica agli amministratori le modifiche allo stato dei dispositivi principali Greengrass. Ciò consente di identificare un comportamento anomalo che potrebbe indicare la compromissione del dispositivo. Per ulteriori informazioni, consulta la sezione AWS IoT Device Defender nella Guida per gli sviluppatori di AWS IoT Core .

Questo componente legge le metriche di sistema sul dispositivo principale. Quindi, pubblica le metriche su. AWS IoT Device Defender Per ulteriori informazioni su come leggere e interpretare

le metriche riportate da questo componente, consulta la sezione Specifiche del documento Device Metrics nella Developer Guide. AWS IoT Core

> **Note**
>
> Questo componente fornisce funzionalità simili al connettore Device Defender in. AWS IoT Greengrass V1 Per ulteriori informazioni, consulta Device Defender connector nella Guida per gli AWS IoT Greengrass V1 sviluppatori.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 3.1.x
- 3.0.x
- 2,0x

Per informazioni sulle modifiche apportate a ciascuna versione del componente, consulta il changelog.

Type

v3.x

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

v2.x

Questo componente è un componente Lambda () `aws.greengrass.lambda`. Il nucleo Greengrass esegue la funzione Lambda di questo componente utilizzando il componente di avvio Lambda.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

v3.x

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

v2.x

Questo componente può essere installato solo sui dispositivi principali di Linux.

Requisiti

Questo componente ha i seguenti requisiti:

v3.x

- Python versione 3.7 installata sul dispositivo principale e aggiunta alla variabile di ambiente PATH.
- AWS IoT Device Defender configurato per utilizzare la funzione Detect per monitorare le violazioni. Per ulteriori informazioni, consulta Detect nella AWS IoT Core Developer Guide.

v2.x

- Il dispositivo principale deve soddisfare i requisiti per eseguire le funzioni Lambda. Se desideri che il dispositivo principale esegua funzioni Lambda containerizzate, il dispositivo deve soddisfare i requisiti per farlo. Per ulteriori informazioni, consulta Requisiti della funzione Lambda.
- Python versione 3.7 installata sul dispositivo principale e aggiunta alla variabile di ambiente PATH.
- AWS IoT Device Defender configurato per utilizzare la funzione Detect per monitorare le violazioni. Per ulteriori informazioni, consulta Detect nella AWS IoT Core Developer Guide.
- La libreria psutil installata sul dispositivo principale. La versione 5.7.0 è l'ultima versione verificata per funzionare con il componente.
- La libreria cbor installata sul dispositivo principale. La versione 1.0.0 è l'ultima versione verificata per funzionare con il componente.
- Per ricevere i dati di output da questo componente, è necessario unire il seguente aggiornamento di configurazione per il componente legacy del router di abbonamento (`aws.greengrass.LegacySubscriptionRouter`) quando si distribuisce questo componente. Questa configurazione specifica l'argomento in cui questo componente pubblica le risposte.

  Legacy subscription router v2.1.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-device-defender": {
        "id": "aws-greengrass-device-defender",
        "source": "component:aws.greengrass.DeviceDefender",
        "subject": "$aws/things/+/defender/metrics/json",
        "target": "cloud"
      }
    }
  }
  ```

  Legacy subscription router v2.0.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-device-defender": {
        "id": "aws-greengrass-device-defender",
  ```

```
      "source": "arn:aws:lambda:region:aws:function:aws-greengrass-device-
defender:version",
      "subject": "$aws/things/+/defender/metrics/json",
      "target": "cloud"
    }
  }
}
```

- Sostituisci la *regione* con quella Regione AWS che usi.
- Sostituisci la *versione* con la versione della funzione Lambda eseguita da questo componente. Per trovare la versione della funzione Lambda, è necessario visualizzare la ricetta per la versione di questo componente che si desidera distribuire. Apri la pagina dei dettagli di questo componente nella AWS IoT Greengrass console e cerca la coppia chiave-valore della funzione Lambda. Questa coppia chiave-valore contiene il nome e la versione della funzione Lambda.

> **Important**
>
> È necessario aggiornare la versione della funzione Lambda sul router di sottoscrizione legacy ogni volta che si distribuisce questo componente. Ciò garantisce l'utilizzo della versione corretta della funzione Lambda per la versione del componente che si distribuisce.

Per ulteriori informazioni, consulta Creare distribuzione.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

3.1.1

La tabella seguente elenca le dipendenze per la versione 3.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <3.0.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

3.0.0 - 3.0.2

La tabella seguente elenca le dipendenze per le versioni da 3.0.0 a 3.0.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <3.0.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.0.10 and 2.0.11

La tabella seguente elenca le dipendenze per le versioni 2.0.10 e 2.0.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.9

La tabella seguente elenca le dipendenze per la versione 2.0.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Rigidi

Dipendenza	Versioni compatibili	Tipo di dipendenza
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.8

La tabella seguente elenca le dipendenze per la versione 2.0.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.7

La tabella seguente elenca le dipendenze per la versione 2.0.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.6

La tabella seguente elenca le dipendenze per la versione 2.0.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.5

La tabella seguente elenca le dipendenze per la versione 2.0.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.4

La tabella seguente elenca le dipendenze per la versione 2.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.3

La tabella seguente elenca le dipendenze per la versione 2.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Rigidi
Lanciatore Lambda	>=1.0.0	Rigidi
Runtime Lambda	>=1.0.0	Flessibili
Servizio di scambio di token	>=1.0.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

v3.x

`PublishRetryCount`

Il numero di volte in cui la pubblicazione verrà ritentata. Questa funzionalità è disponibile nella versione 3.1.1.

Il valore minimo è 0.

Il massimo è 72.

Impostazione predefinita: 5

`SampleIntervalSeconds`

(Facoltativo) La quantità di tempo in secondi tra ogni ciclo in cui il componente raccoglie e riporta le metriche.

Il valore minimo è di 300 secondi (5 minuti).

Impostazione predefinita: 300 secondi

`UseInstaller`

(Facoltativo) Valore booleano che definisce se utilizzare lo script di installazione in questo componente per installarne le dipendenze.

Imposta questo valore su `false` se desideri utilizzare uno script personalizzato per installare le dipendenze o se desideri includere le dipendenze di runtime in un'immagine Linux predefinita. Per utilizzare questo componente, è necessario installare le seguenti librerie, comprese le eventuali dipendenze, e renderle disponibili all'utente predefinito del sistema Greengrass.

- SDK per dispositivi AWS IoT v2 per Python
- libreria cbor. La versione 1.0.0 è l'ultima versione verificata per funzionare con il componente.
- libreria psutil. La versione 5.7.0 è l'ultima versione verificata per funzionare con il componente.

> **Note**
>
> Se si utilizza la versione 3.0.0 o 3.0.1 di questo componente su dispositivi principali configurati per l'utilizzo di un proxy HTTPS, è necessario impostare questo valore su. `false` Lo script di installazione non supporta il funzionamento con un proxy HTTPS in queste versioni di questo componente.

Impostazione predefinita: `true`

v2.x

> **Note**
>
> La configurazione predefinita di questo componente include i parametri della funzione Lambda. Ti consigliamo di modificare solo i seguenti parametri per configurare questo componente sui tuoi dispositivi.

`lambdaParams`

Un oggetto che contiene i parametri per la funzione Lambda di questo componente. Questo oggetto contiene le seguenti informazioni:

`EnvironmentVariables`

Un oggetto che contiene i parametri della funzione Lambda. Questo oggetto contiene le seguenti informazioni:

`PROCFS_PATH`

(Facoltativo) Il percorso della `/proc` cartella.

- Per eseguire questo componente in un contenitore, utilizzate il valore predefinito,`/host-proc`. Per impostazione predefinita, il componente viene eseguito in un contenitore.
- Per eseguire questo componente in modalità senza contenitore, specificare `/proc` questo parametro.

Default: `/host-proc`. Questo è il percorso predefinito in cui questo componente monta la `/proc` cartella nel contenitore.

> **Note**
>
> Questo componente ha accesso in sola lettura a questa cartella.

`SAMPLE_INTERVAL_SECONDS`

(Facoltativo) La quantità di tempo in secondi tra ogni ciclo in cui il componente raccoglie e riporta le metriche.

Il valore minimo è di 300 secondi (5 minuti).

Impostazione predefinita: 300 secondi

`containerMode`

(Facoltativo) La modalità di containerizzazione per questo componente. Seleziona una delle opzioni seguenti:

- `GreengrassContainer`— Il componente viene eseguito in un ambiente di runtime isolato all'interno del AWS IoT Greengrass contenitore.

- `NoContainer`— Il componente non viene eseguito in un ambiente di runtime isolato.

  Se si specifica questa opzione, è necessario specificare `/proc` il parametro della variabile di `PROCFS_PATH` ambiente.

Impostazione predefinita: `GreengrassContainer`

`containerParams`

(Facoltativo) Un oggetto che contiene i parametri del contenitore per questo componente. Il componente utilizza questi parametri se si specifica `GreengrassContainer` per`containerMode`.

Questo oggetto contiene le seguenti informazioni:

`memorySize`

(Facoltativo) La quantità di memoria (in kilobyte) da allocare al componente.

Il valore predefinito è 50.000 KB.

`pubsubTopics`

(Facoltativo) Un oggetto che contiene gli argomenti a cui il componente si iscrive per ricevere messaggi. È possibile specificare ogni argomento e se il componente sottoscrive gli argomenti MQTT AWS IoT Core o gli argomenti di pubblicazione/sottoscrizione locali.

Questo oggetto contiene le seguenti informazioni:

`0`— Si tratta di un indice di matrice sotto forma di stringa.

Un oggetto che contiene le seguenti informazioni:

`type`

(Facoltativo) Il tipo di messaggi di pubblicazione/sottoscrizione utilizzato da questo componente per sottoscrivere i messaggi. Seleziona una delle opzioni seguenti:

- `PUB_SUB`: iscriviti ai messaggi di pubblicazione/sottoscrizione locali. Se scegli questa opzione, l'argomento non può contenere caratteri jolly MQTT. Per ulteriori informazioni su come inviare messaggi dal componente personalizzato quando si specifica questa opzione, vedere. Pubblicare/sottoscrivere messaggi locali
- `IOT_CORE`— Abbonarsi ai messaggi AWS IoT Core MQTT. Se scegli questa opzione, l'argomento può contenere caratteri jolly MQTT. Per ulteriori informazioni su come

inviare messaggi da componenti personalizzati quando si specifica questa opzione, vedere. AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

Impostazione predefinita: PUB_SUB

`topic`

(Facoltativo) L'argomento a cui il componente si iscrive per ricevere messaggi. Se si specifica `IotCore` for`type`, è possibile utilizzare i caratteri jolly MQTT (+and#) in questo argomento.

Example Esempio: aggiornamento basato sull'unione della configurazione (modalità contenitore)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "PROCFS_PATH": "/host_proc"
    }
  },
  "containerMode": "GreengrassContainer"
}
```

Example Esempio: aggiornamento tramite fusione della configurazione (nessuna modalità contenitore)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "PROCFS_PATH": "/proc"
    }
  },
  "containerMode": "NoContainer"
}
```

Dati di input

Questo componente non accetta messaggi come dati di input.

Dati di output

Questo componente pubblica le metriche di sicurezza relative al seguente argomento riservato a. AWS IoT Device Defender Questo componente lo sostituisce *coreDeviceName* con il nome del dispositivo principale quando pubblica le metriche.

Argomento (MQTT)AWS IoT Core : `$aws/things/`*coreDeviceName*`/defender/metrics/json`

Example Output di esempio

```
{
  "header": {
    "report_id": 1529963534,
    "version": "1.0"
  },
  "metrics": {
    "listening_tcp_ports": {
      "ports": [
        {
          "interface": "eth0",
          "port": 24800
        },
        {
          "interface": "eth0",
          "port": 22
        },
        {
          "interface": "eth0",
          "port": 53
        }
      ],
      "total": 3
    },
    "listening_udp_ports": {
      "ports": [
        {
          "interface": "eth0",
          "port": 5353
        },
        {
          "interface": "eth0",
          "port": 67
        }
```

```
      ],
      "total": 2
    },
    "network_stats": {
      "bytes_in": 1157864729406,
      "bytes_out": 1170821865,
      "packets_in": 693092175031,
      "packets_out": 738917180
    },
    "tcp_connections": {
      "established_connections":{
        "connections": [
          {
            "local_interface": "eth0",
            "local_port": 80,
            "remote_addr": "192.168.0.1:8000"
          },
          {
            "local_interface": "eth0",
            "local_port": 80,
            "remote_addr": "192.168.0.1:8000"
          }
        ],
        "total": 2
      }
    }
  }
}
```

Per ulteriori informazioni sulle metriche riportate da questo componente, consulta la sezione Specifiche del documento Device Metrics nella Developer Guide. AWS IoT Core

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.DeviceDefender.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.DeviceDefender.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.greengrass.DeviceDefender.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.DeviceDefender.log -Tail 10 -
  Wait
  ```

Licenze

Questo componente è rilasciato in base al contratto di licenza del software Greengrass Core.

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

v3.x

Versione	Modifiche
3.1.1	Correzioni di bug e miglioramenti • Aggiunge nuovi tentativi di connessione al client quando la connessio ne non riesce a ripristinarsi dopo un'interruzione della rete. • Aggiunge un nuovo tentativo configurabile per la pubblicazione delle metriche.

Versione	Modifiche
3.1.0	Correzioni di bug e miglioramenti • Aggiunge il supporto per le configurazioni proxy di rete HTTPS. Per ulteriori informazioni, consultare Connessione alla porta 443 o tramite un proxy di rete e Abilita il dispositivo principale in modo che consideri attendibile un proxy HTTPS.
3.0.1	Risolve un problema relativo al modo in cui il componente calcola i valori delta per le metriche.
3.0.0	**Warning** Questa versione non è più disponibile. I miglioramenti di questa versione sono disponibili nelle versioni successive di questo componente. Versione iniziale.

v2.x

Versione	Modifiche
2.0.11	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.0.10	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.0.9	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.0.8	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.0.7	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.6	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.0.5	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.

Versione	Modifiche
2.0.4	Versione aggiornata per la versione 2.1.0 di Greengrass nucleus.
2.0.3	Versione iniziale.

Spooler del disco

Il componente disk spooler (`aws.greengrass.DiskSpooler`) offre un'opzione di archiviazione persistente per i messaggi trasferiti dai dispositivi core di Greengrass a. AWS IoT Core Questo componente memorizzerà questi messaggi in uscita su disco.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 1.0.x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- `storageType`deve essere impostato `Disk` per utilizzare questo componente. Puoi impostarlo nella configurazione Greengrass nucleus.
- `maxSizeInBytes`non deve essere configurato per occupare una dimensione superiore allo spazio disponibile sul dispositivo. Puoi impostarlo nella configurazione Greengrass nucleus.
- Il componente disk spooler è supportato per l'esecuzione in un VPC.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

1.0.1 – 1.0.3

La tabella seguente elenca le dipendenze per le versioni da 1.0.1 a 1.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.11.0 <2.13.0	Rigidi

1.0.0

La tabella seguente elenca le dipendenze per la versione 1.0.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.11.0 <2.12.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Utilizzo

Per utilizzare il componente disk spooler, `aws.greengrass.DiskSpooler` deve essere distribuito.

Per configurare e utilizzare questo componente, è necessario impostare su. `pluginName` `aws.greengrass.DiskSpooler`

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
```

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
1.0.3	Correzioni di bug e miglioramenti Migliora le prestazioni riutilizzando le connessioni al database.
1.0.2	Correzioni di bug e miglioramenti Risolve un problema per cui il campo del formato del messaggio MQTT non veniva mantenuto in alcuni casi.
1.0.1	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
1.0.0	Versione iniziale.

Gestore di applicazioni Docker

Il componente Docker application manager (`aws.greengrass.DockerApplicationManager`) consente AWS IoT Greengrass di scaricare immagini Docker da registri di immagini pubblici e registri privati ospitati su Amazon Elastic Container Registry (Amazon ECR) Elastic Container Registry (Amazon ECR). Consente inoltre di AWS IoT Greengrass gestire automaticamente le credenziali per scaricare in modo sicuro immagini da archivi privati in Amazon ECR.

Quando sviluppi un componente personalizzato che esegue un contenitore Docker, includi il gestore delle applicazioni Docker come dipendenza per scaricare le immagini Docker specificate come artefatti nel componente. Per ulteriori informazioni, consulta Esegui un contenitore Docker.

Argomenti

- Versioni

Versioni

Questo componente ha le seguenti versioni:

- 2.0.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Docker Engine 1.9.1 o versione successiva installato sul dispositivo principale Greengrass. La versione 20.10 è l'ultima versione verificata per funzionare con il software Core. AWS IoT

Greengrass È necessario installare Docker direttamente sul dispositivo principale prima di distribuire componenti che eseguono contenitori Docker.

- Il daemon Docker è stato avviato e funzionante sul dispositivo principale prima di distribuire questo componente.
- Immagini Docker archiviate in una delle seguenti fonti di immagini supportate:
  - Archivi di immagini pubblici e privati in Amazon Elastic Container Registry (Amazon ECR)
  - Archivio pubblico di Docker Hub
  - Registro pubblico affidabile di Docker
- Immagini Docker incluse come artefatti nei componenti del contenitore Docker personalizzati. Usa i seguenti formati URI per specificare le tue immagini Docker:
  - Immagine Amazon ECR privata: `docker:`*`account-id`*`.dkr.ecr.`*`region`*`.amazonaws.com/`*`repository`*`/`*`image`*`[`*`:tag|@digest`*`]`
  - Immagine pubblica di Amazon ECR: `docker:public.ecr.aws/`*`repository`*`/`*`image`*`[`*`:tag|@digest`*`]`
  - Immagine pubblica di Docker Hub: `docker:`*`name`*`[`*`:tag|@digest`*`]`

  Per ulteriori informazioni, consulta Esegui un contenitore Docker.

  > **Note**
  >
  > Se non specifichi il tag dell'immagine o l'image digest nell'URI dell'artefatto per un'immagine, il gestore delle applicazioni Docker recupera l'ultima versione disponibile di quell'immagine quando distribuisci il componente contenitore Docker personalizzato. Per garantire che tutti i dispositivi principali eseguano la stessa versione di un'immagine, ti consigliamo di includere il tag dell'immagine o l'image digest nell'URI dell'artefatto.

- L'utente di sistema che esegue un componente del contenitore Docker deve disporre delle autorizzazioni di root o amministratore oppure è necessario configurare Docker per eseguirlo come utente non root o non amministratore.
  - Sui dispositivi Linux, puoi aggiungere un utente al gruppo senza il quale chiamare i comandi. `docker` `docker` `sudo`
  - Nei dispositivi Windows, è possibile aggiungere un utente al `docker-users` gruppo per richiamare `docker` comandi senza privilegi di amministratore.

Linux or Unix

Per aggiungere `ggc_user` al `docker` gruppo l'utente non root che usi per eseguire i componenti del contenitore Docker, esegui il comando seguente.

```
sudo usermod -aG docker ggc_user
```

Per ulteriori informazioni, consulta Gestire Docker come utente non root.

Windows Command Prompt (CMD)

Per aggiungere al `docker-users` gruppo`ggc_user`, o l'utente che usi per eseguire i componenti del contenitore Docker, esegui il comando seguente come amministratore.

```
net localgroup docker-users ggc_user /add
```

Windows PowerShell

Per aggiungere al `docker-users` gruppo`ggc_user`, o l'utente che usi per eseguire i componenti del contenitore Docker, esegui il comando seguente come amministratore.

```
Add-LocalGroupMember -Group docker-users -Member ggc_user
```

- Se configuri il software AWS IoT Greengrass Core per utilizzare un proxy di rete, devi configurare Docker per utilizzare lo stesso server proxy.
- Se le tue immagini Docker sono archiviate in un registro privato Amazon ECR, devi includere il componente del servizio di scambio di token come dipendenza nel componente contenitore Docker. Inoltre, il ruolo del dispositivo Greengrass deve consentire le `ecr:GetDownloadUrlForLayer` azioni`ecr:GetAuthorizationToken`, e`ecr:BatchGetImage`, come mostrato nell'esempio seguente di politica IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ],
```

```
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

- Il componente docker application manager è supportato per l'esecuzione in un VPC. Per distribuire questo componente in un VPC, è necessario quanto segue.
  - Il componente docker application manager deve disporre di connettività per scaricare le immagini. Ad esempio, se si utilizza ECR, è necessario disporre della connettività ai seguenti endpoint.
    - `*.dkr.ecr.region.amazonaws.com`(endpoint VPC) `com.amazonaws.region.ecr.dkr`
    - `api.ecr.region.amazonaws.com`(endpoint VPC) `com.amazonaws.region.ecr.api`

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`ecr.region.amazonaws.com`	443	No	Obbligatorio se scarichi immagini Docker da Amazon ECR.

`hub.docker.com`
 	
`registry.hub.docker.com/v1`
	443	No	Obbligatorio se scarichi immagini Docker

Endpoint	Porta	Richiesto	Descrizione
			da Docker Hub.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.0.11

La tabella seguente elenca le dipendenze per la versione 2.0.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.13.0	Flessibili

2.0.10

La tabella seguente elenca le dipendenze per la versione 2.0.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.12.0	Flessibili

2.0.9

La tabella seguente elenca le dipendenze per la versione 2.0.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.11.0	Flessibili

2.0.8

La tabella seguente elenca le dipendenze per la versione 2.0.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.10.0	Flessibili

2.0.7

La tabella seguente elenca le dipendenze per la versione 2.0.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.9.0	Flessibili

2.0.6

La tabella seguente elenca le dipendenze per la versione 2.0.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.8.0	Flessibili

2.0.5

La tabella seguente elenca le dipendenze per la versione 2.0.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.7.0	Flessibili

2.0.4

La tabella seguente elenca le dipendenze per la versione 2.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.6.0	Flessibili

2.0.3

La tabella seguente elenca le dipendenze per la versione 2.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.5.0	Flessibili

2.0.2

La tabella seguente elenca le dipendenze per la versione 2.0.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.4.0	Flessibili

2.0.1

La tabella seguente elenca le dipendenze per la versione 2.0.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.3.0	Flessibili

2.0.0

La tabella seguente elenca le dipendenze per la versione 2.0.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.2.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente non ha parametri di configurazione.

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.0.11	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.0.10	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.0.9	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.0.8	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.0.7	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.0.6	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.0.5	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.0.4	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.0.3	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.2	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.0.1	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.0.0	Versione iniziale.

Consulta anche

- Esegui un contenitore Docker

Connettore Edge per Kinesis Video Streams

Il connettore perimetrale per il `aws.iot.EdgeConnectorForKVS` componente Kinesis Video Streams () legge i feed video dalle telecamere locali e li pubblica su Kinesis Video Streams. È possibile configurare questo componente per leggere i feed video dalle telecamere IP (Internet Protocol) utilizzando il protocollo RTSP (Real Time Streaming Protocol). Quindi, puoi configurare

dashboard in Amazon Managed Grafana o server Grafana locali per monitorare e interagire con i flussi video.

Puoi integrare questo componente con AWS IoT TwinMaker per visualizzare e controllare i flussi video nelle dashboard Grafana. AWS IoT TwinMaker è un AWS servizio che consente di creare gemelli digitali operativi di sistemi fisici. È possibile utilizzarlo AWS IoT TwinMaker per visualizzare i dati provenienti da sensori, fotocamere e applicazioni aziendali per monitorare le fabbriche fisiche, gli edifici o gli impianti industriali. È inoltre possibile utilizzare questi dati per monitorare le operazioni, diagnosticare errori e correggere errori. Per ulteriori informazioni, consulta Che cos'è AWS IoT TwinMaker? nella Guida per l'utente di AWS IoT TwinMaker .

Questo componente memorizza la sua configurazione inAWS IoT SiteWise, un AWS servizio che modella e archivia i dati industriali. InAWS IoT SiteWise, le risorse rappresentano oggetti come dispositivi, apparecchiature o gruppi di altri oggetti. Per configurare e utilizzare questo componente, crei una AWS IoT SiteWise risorsa per ogni dispositivo principale Greengrass e per ogni telecamera IP collegata a ciascun dispositivo principale. Ogni risorsa ha proprietà che configurate per controllare funzionalità, come lo streaming live, il caricamento su richiesta e la memorizzazione nella cache locale. Per specificare l'URL di ogni videocamera, create una cartella segreta AWS Secrets Manager che contiene l'URL della videocamera. Se la fotocamera richiede l'autenticazione, specificate anche un nome utente e una password nell'URL. Quindi, specificate quel segreto in una proprietà della risorsa per la telecamera IP.

Questo componente carica il flusso video di ogni telecamera su un flusso video Kinesis. È necessario specificare il nome del flusso video Kinesis di destinazione nella configurazione degli AWS IoT SiteWise asset per ogni telecamera. Se lo stream video Kinesis non esiste, questo componente lo crea automaticamente.

AWS IoT TwinMaker fornisce uno script che è possibile eseguire per creare queste AWS IoT SiteWise risorse e i segreti di Secrets Manager. Per ulteriori informazioni su come creare queste risorse e su come installare, configurare e utilizzare questo componente, consultate l'integrazione AWS IoT TwinMaker video nella Guida per l'AWS IoT TwinMaker utente.

> **Note**
>
> Il connettore perimetrale per il componente Kinesis Video Streams è disponibile solo nei seguenti casi: Regioni AWS
>
> - Stati Uniti orientali (Virginia settentrionale)
> - US West (Oregon)

- Europa (Francoforte)
- Europa (Irlanda)
- Asia Pacifico (Singapore)

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 1.0.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato solo sui dispositivi principali di Linux.

Requisiti

Questo componente ha i seguenti requisiti:

- È possibile distribuire questo componente solo su dispositivi single-core, poiché la configurazione del componente deve essere unica per ogni dispositivo principale. Non puoi distribuire questo componente su gruppi di dispositivi principali.
- GStreamer 1.18.4 o versione successiva installato sul dispositivo principale. Per ulteriori informazioni, consulta Installazione di GStreamer.

  Su un dispositivo con `apt`, puoi eseguire i seguenti comandi per installare GStreamer.

  ```
  sudo apt install -y libgstreamer1.0-dev libgstreamer-plugins-base1.0-dev
   gstreamer1.0-plugins-base-apps
  sudo apt install -y gstreamer1.0-libav
  sudo apt install -y gstreamer1.0-plugins-bad gstreamer1.0-plugins-good gstreamer1.0-
  plugins-ugly gstreamer1.0-tools
  ```

- Una AWS IoT SiteWise risorsa per ogni dispositivo principale. Questa AWS IoT SiteWise risorsa rappresenta il dispositivo principale. Per ulteriori informazioni su come creare questa risorsa, consultate l'integrazione AWS IoT TwinMaker video nella Guida per l'AWS IoT TwinMaker utente.
- Una AWS IoT SiteWise risorsa per ogni telecamera IP collegata a ciascun dispositivo principale. Queste AWS IoT SiteWise risorse rappresentano le telecamere che trasmettono video a ciascun dispositivo principale. La risorsa di ogni videocamera deve essere associata alla risorsa per il dispositivo principale che si collega alla videocamera. Le risorse della videocamera hanno proprietà che puoi configurare per specificare un flusso video Kinesis, un segreto di autenticazione e parametri di streaming video. Per ulteriori informazioni su come creare e configurare le risorse della videocamera, consulta l'integrazione AWS IoT TwinMaker video nella Guida per l'AWS IoT TwinMaker utente.
- Un AWS Secrets Manager segreto per ogni telecamera IP. Questo segreto deve definire una coppia chiave-valore, dove si trova la chiave e il valore è l'URL della telecamera. `RTSPStreamUrl` Se la fotocamera richiede l'autenticazione, includi il nome utente e la password in questo URL. È possibile utilizzare uno script per creare un segreto quando si creano le risorse richieste da questo componente. Per ulteriori informazioni, consulta l'integrazione AWS IoT TwinMaker video nella Guida AWS IoT TwinMaker per l'utente.

  Puoi anche utilizzare la console e l'API di Secrets Manager per creare segreti aggiuntivi. Per ulteriori informazioni, consulta Creare un segreto nella Guida AWS Secrets Manager per l'utente.

- Il ruolo di scambio di token Greengrass deve consentire le seguenti azioni e quelle di Kinesis Video Streams AWS Secrets ManagerAWS IoT SiteWise, come illustrato nell'esempio seguente di policy IAM.

> **Note**
>
> Questa policy di esempio consente al dispositivo di ottenere il valore dei segreti denominati and. **`IPCamera1Url` `IPCamera2Url`** Quando si configura ogni telecamera IP, si specifica un segreto che contiene l'URL di quella telecamera. Se la telecamera richiede l'autenticazione, si specificano anche un nome utente e una password nell'URL. Il ruolo di scambio di token del dispositivo principale deve consentire l'accesso al segreto per ogni telecamera IP da connettere.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:secretsmanager:region:account-id:secret:IPCamera1Url",
        "arn:aws:secretsmanager:region:account-id:secret:IPCamera2Url"
      ]
    },
    {
      "Action": [
        "iotsitewise:BatchPutAssetPropertyValue",
        "iotsitewise:DescribeAsset",
        "iotsitewise:DescribeAssetModel",
        "iotsitewise:DescribeAssetProperty",
        "iotsitewise:GetAssetPropertyValue",
        "iotsitewise:ListAssetRelationships",
        "iotsitewise:ListAssets",
        "iotsitewise:ListAssociatedAssets",
        "kinesisvideo:CreateStream",
        "kinesisvideo:DescribeStream",
        "kinesisvideo:GetDataEndpoint",
        "kinesisvideo:PutMedia",
```

```
        "kinesisvideo:TagStream"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    }
  ]
}
```

> **Note**
>
> Se si utilizza una AWS Key Management Service chiave gestita dal cliente per crittografare i segreti, anche il ruolo del dispositivo deve consentire l'`kms:Decrypt`azione.

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`kinesisvideo.` *`region`*`.amazonaws.com`	443	Sì	Carica i dati su Kinesis Video Streams.
`data.iotsitewise.` *`region`*`.amazonaws.com`	443	Sì	Pubblica i metadati dei flussi video su. AWS IoT SiteWise

Endpoint	Porta	Richiesto	Descrizione
`secretsmanager. region.amazonaws.com`	443	Sì	Scarica i segreti degli URL della fotocamera sul dispositivo principale.

Dipendenze

Quando distribuisci un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle sue dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

La tabella seguente elenca le dipendenze per le versioni da 1.0.0 a 1.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Servizio di scambio di token	>=2.0.3	Rigidi
Gestore di stream	>=2.0.9	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`SiteWiseAssetIdForHub`

L'ID della AWS IoT SiteWise risorsa che rappresenta questo dispositivo principale. Per ulteriori informazioni su come creare questa risorsa e utilizzarla per interagire con questo componente, consultate l'integrazione AWS IoT TwinMaker video nella Guida per l'AWS IoT TwinMaker utente.

Example Esempio: fusione e aggiornamento della configurazione

```
{
  "SiteWiseAssetIdForHub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
}
```

Licenze

Questo componente include i seguenti software/licenze di terze parti:

- Quartz Job Scheduler/ Licenza Apache 2.0
- Collegamenti Java per GStreamer 1.x /GNU Lesser General Public License v3.0

Utilizzo

Per configurare e interagire con questo componente, puoi impostare le proprietà degli AWS IoT SiteWise asset che rappresentano il dispositivo principale e le telecamere IP a cui si connette. Puoi anche visualizzare e interagire con i flussi video nelle dashboard di Grafana tramite. AWS IoT TwinMaker Per ulteriori informazioni, consulta l'integrazione AWS IoT TwinMaker video nella Guida per l'utente. AWS IoT TwinMaker

File di registro locale

Questo componente utilizza il seguente file di registro.

```
/greengrass/v2/logs/aws.iot.EdgeConnectorForKVS.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale.

```
sudo tail -f /greengrass/v2/logs/aws.iot.EdgeConnectorForKVS.log
```

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
1.0.4	Correzioni di bug e miglioramenti • Risolve un problema che causava l'interruzione del caricamento dal vivo.
1.0.3	Correzioni di bug generali e miglioramenti.
1.0.1	Correzioni di bug generali e miglioramenti.
1.0.0	Versione iniziale.

Consulta anche

- Cos'è AWS IoT TwinMaker? nella Guida per l'utente di AWS IoT TwinMaker
- AWS IoT TwinMaker integrazione video nella Guida per l'utente AWS IoT TwinMaker
- Cos'è AWS IoT SiteWise? nella Guida per l'utente di AWS IoT SiteWise
- Aggiornamento dei valori degli attributi nella Guida AWS IoT SiteWise per l'utente
- Cos'è AWS Secrets Manager? nella Guida per l'utente di AWS Secrets Manager
- Crea e gestisci segreti nella Guida AWS Secrets Manager per l'utente

Greengrass CLI

Il componente Greengrass CLI (`aws.greengrass.Cli`) fornisce un'interfaccia a riga di comando locale che è possibile utilizzare sui dispositivi principali per sviluppare ed eseguire il debug dei componenti localmente. La CLI Greengrass consente, ad esempio, di creare distribuzioni locali e riavviare i componenti sul dispositivo principale.

È possibile installare questo componente quando si installa il software Core. AWS IoT Greengrass Per ulteriori informazioni, consulta Tutorial: Nozioni di base su AWS IoT Greengrass V2.

> **Important**
>
> Si consiglia di utilizzare questo componente solo in ambienti di sviluppo, non in ambienti di produzione. Questo componente fornisce l'accesso a informazioni e operazioni che in genere non sono necessarie in un ambiente di produzione. Segui il principio del privilegio minimo distribuendo questo componente solo sui dispositivi principali dove ne hai bisogno.

Dopo aver installato questo componente, esegui il comando seguente per visualizzarne la documentazione di aiuto. Quando questo componente viene installato, aggiunge un collegamento simbolico alla `greengrass-cli` */greengrass/v2*`/bin` cartella. È possibile eseguire la CLI Greengrass da questo percorso o aggiungerla alla variabile di `PATH` ambiente per eseguirla `greengrass-cli` senza il percorso assoluto.

Linux or Unix

```
/greengrass/v2/bin/greengrass-cli help
```

Windows

```
C:\greengrass\v2\bin\greengrass-cli help
```

Il comando seguente riavvia un componente denominato`com.example.HelloWorld`, ad esempio.

Linux or Unix

```
sudo /greengrass/v2/bin/greengrass-cli component restart --names
 "com.example.HelloWorld"
```

Windows

```
C:\greengrass\v2\bin\greengrass-cli component restart --names
 "com.example.HelloWorld"
```

Per ulteriori informazioni, consulta Interfaccia a riga di comando Greengrass.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.12.x
- 2.11. x
- 2.10.x
- 2.9. x
- 2.8.x
- 2.7.x
- 2.6. x
- 2,5. x
- 2.4.x
- 2.3.x
- 2.2.x
- 2.1.x
- 2,0x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- È necessario essere autorizzati a utilizzare la CLI Greengrass per interagire con il software Core. AWS IoT Greengrass Effettuate una delle seguenti operazioni per utilizzare la Greengrass CLI:
  - Utilizzate l'utente di sistema che esegue il software AWS IoT Greengrass Core.
  - Usa un utente con autorizzazioni root o amministrative. Sui dispositivi principali di Linux, puoi utilizzarlo per ottenere i permessi `sudo` di root.
  - Usa un utente di sistema che fa parte di un gruppo specificato nei parametri di `AuthorizedWindowsGroups` configurazione `AuthorizedPosixGroups` o quando distribuisci il componente. Per ulteriori informazioni, consulta Configurazione dei componenti della CLI di Greengrass.
- Il componente Greengrass CLI è supportato per l'esecuzione in un VPC.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle sue dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.12.0 – 2.12.4

La tabella seguente elenca le dipendenze per le versioni da 2.12.0 a 2.12.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.12.0 <2.13.0	Flessibili

2.11.0 – 2.11.3

La tabella seguente elenca le dipendenze per le versioni da 2.11.0 a 2.11.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.11.0 <2.12.0	Flessibili

2.10.0 – 2.10.3

La tabella seguente elenca le dipendenze per le versioni da 2.10.0 a 2.10.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.11.0	Flessibili

2.9.0 – 2.9.6

La tabella seguente elenca le dipendenze per le versioni da 2.9.0 a 2.9.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.10.0	Flessibili

2.8.0 – 2.8.1

La tabella seguente elenca le dipendenze per le versioni 2.8.0 e 2.8.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.9.0	Flessibili

2.7.0

La tabella seguente elenca le dipendenze per la versione 2.7.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.8.0	Flessibili

2.6.0

La tabella seguente elenca le dipendenze per la versione 2.6.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.7.0	Flessibili

2.5.0 – 2.5.6

La tabella seguente elenca le dipendenze per le versioni da 2.5.0 a 2.5.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.6.0	Flessibili

2.4.0

La tabella seguente elenca le dipendenze per la versione 2.4.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.5.0	Flessibili

2.3.0

La tabella seguente elenca le dipendenze per la versione 2.3.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.4.0	Flessibili

2.2.0

La tabella seguente elenca le dipendenze per la versione 2.2.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.3.0	Flessibili

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.2.0	Flessibili

2.0.x

La tabella seguente elenca le dipendenze per la versione 2.0.x di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.1.0	Flessibili

> **Note**
>
> La versione minima compatibile del nucleo Greengrass corrisponde alla versione patch del componente Greengrass CLI.

Per ulteriori informazioni sulle dipendenze dei componenti, consultate il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

2.5.x

`AuthorizedPosixGroups`

(Facoltativo) Una stringa che contiene un elenco di gruppi di sistema separati da virgole. Autorizzi questi gruppi di sistema a utilizzare la CLI Greengrass per interagire con AWS IoT Greengrass il software Core. È possibile specificare nomi o ID di gruppo. Ad esempio, `group1,1002,group3` autorizza tre gruppi di sistema (`group1`1002, e`group3`) a utilizzare la CLI Greengrass.

Se non specifichi alcun gruppo da autorizzare, puoi utilizzare la CLI Greengrass come `sudo` utente root () o come utente di sistema che AWS IoT Greengrass esegue il software Core.

`AuthorizedWindowsGroups`

(Facoltativo) Una stringa che contiene un elenco separato da virgole di gruppi di sistema. Autorizzi questi gruppi di sistema a utilizzare la CLI Greengrass per interagire con AWS IoT Greengrass il software Core. È possibile specificare nomi o ID di gruppo. Ad esempio, `group1,1002,group3` autorizza tre gruppi di sistema (`group1`1002, e`group3`) a utilizzare la CLI Greengrass.

Se non specifichi alcun gruppo da autorizzare, puoi utilizzare la CLI Greengrass come amministratore o come utente di sistema che AWS IoT Greengrass esegue il software Core.

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica di autorizzare tre gruppi di sistema POSIX (`group11002`, `egroup3`) e due gruppi di utenti Windows (`Device OperatorsandQA Engineers`) a utilizzare la Greengrass CLI.

```
{
  "AuthorizedPosixGroups": "group1,1002,group3",
  "AuthorizedWindowsGroups": "Device Operators,QA Engineers"
}
```

2.4.x - 2.0.x

`AuthorizedPosixGroups`

(Facoltativo) Una stringa che contiene un elenco di gruppi di sistema separati da virgole. Autorizzi questi gruppi di sistema a utilizzare la CLI Greengrass per interagire con AWS IoT Greengrass il software Core. È possibile specificare nomi o ID di gruppo. Ad esempio, `group1,1002,group3` autorizza tre gruppi di sistema (`group11002`, `egroup3`) a utilizzare la CLI Greengrass.

Se non specifichi alcun gruppo da autorizzare, puoi utilizzare la CLI Greengrass come `sudo` utente root () o come utente di sistema che AWS IoT Greengrass esegue il software Core.

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica di autorizzare tre gruppi di sistema (`group11002`, `egroup3`) a utilizzare la Greengrass CLI.

```
{
  "AuthorizedPosixGroups": "group1,1002,group3"
}
```

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.12.4	Versione aggiornata per la versione 2.12.4 di Greengrass nucleus.
2.12.3	**Warning** Questa versione non è più disponibile. I miglioramenti di questa versione sono disponibili nelle versioni successive di questo componente. Versione aggiornata per la versione 2.12.3 di Greengrass nucleus.
2.12.2	Versione aggiornata per la versione 2.12.2 di Greengrass nucleus.

Versione	Modifiche
2.12.1	Versione aggiornata per la versione 2.12.1 di Greengrass nucleus.
2.12.0	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.11.3	Versione aggiornata per la versione 2.11.3 di Greengrass nucleus.
2.11.2	Versione aggiornata per la versione 2.11.2 di Greengrass nucleus.
2.11.1	Versione aggiornata per la versione 2.11.1 di Greengrass nucleus.
2.11.0	Nuove funzionalità • Consente di annullare una distribuzione locale. • Consente di configurare una politica di gestione degli errori per una distribuzione locale. • Migliora la segnalazione dettagliata dello stato dell'implementazione.
2.10.3	Versione aggiornata per la versione 2.10.3 di Greengrass nucleus.
2.10.2	Versione aggiornata per la versione 2.10.2 di Greengrass nucleus.
2.10.1	Versione aggiornata per la versione 2.10.1 di Greengrass nucleus.
2.10.0	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.9.6	Versione aggiornata per la versione 2.9.6 di Greengrass nucleus.
2.9.5	Versione aggiornata per la versione 2.9.5 di Greengrass nucleus.
2.9.4	Versione aggiornata per la versione 2.9.4 di Greengrass nucleus.
2.9.3	Versione aggiornata per la versione 2.9.3 di Greengrass nucleus.
2.9.2	Versione aggiornata per la versione 2.9.2 di Greengrass nucleus.
2.9.1	Versione aggiornata per la versione 2.9.1 di Greengrass nucleus.
2.9.0	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.8.1	Versione aggiornata per la versione 2.8.1 di Greengrass nucleus.

Versione	Modifiche
2.8.0	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.7.0	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.6.0	Nuove funzionalità • Aggiunge il supporto per i componenti personalizzati per chiamare le operazioni di comunicazione tra processi (IPC) utilizzate dalla CLI di Greengrass. È possibile utilizzare queste operazioni IPC per gestire le distribuzioni locali, visualizzare i dettagli dei componenti e generare una password da utilizzare per accedere alla console di debug locale. Per ulteriori informazioni, consulta IPC: gestione delle distribuzioni e dei componenti locali. Correzioni di bug e miglioramenti • Correzioni e miglioramenti minori aggiuntivi.
2.5.6	Versione aggiornata per la versione 2.5.6 di Greengrass nucleus.
2.5.5	Versione aggiornata per la versione 2.5.5 di Greengrass nucleus.
2.5.4	Versione aggiornata per la versione 2.5.4 di Greengrass nucleus.
2.5.3	Versione aggiornata per la versione 2.5.3 di Greengrass nucleus.
2.5.2	Versione aggiornata per la versione 2.5.2 di Greengrass nucleus.
2.5.1	Versione aggiornata per la versione 2.5.1 di Greengrass nucleus.
2.5.0	Nuove funzionalità • Aggiunge il supporto per i dispositivi principali che eseguono Windows. • Aggiunge il nuovo parametro di `AuthorizedWindowsGroups` configurazione che è possibile specificare per autorizzare i gruppi di sistema a utilizzare la Greengrass CLI sui dispositivi Windows. • Aggiunge il `windowsUser` parametro per le distribuzioni locali. È possibile utilizzare questo parametro per specificare l'utente da utilizzare per eseguire i componenti su un dispositivo Windows principale.

Versione	Modifiche
2.4.0	Nuove funzionalità • Aggiunge il supporto per i limiti delle risorse di sistema. Quando si crea una distribuzione locale, è possibile configurare la quantità massima di utilizzo di CPU e RAM che i processi di ciascun componente possono utilizzare sul dispositivo principale. Per ulteriori informazioni, consulta Configura i limiti delle risorse di sistema per i componenti e il comando deployment create.
2.3.0	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.2.0	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.1.0	Versione aggiornata per la versione 2.1.0 di Greengrass nucleus.
2.0.5	Versione aggiornata per la versione 2.0.5 di Greengrass nucleus.
2.0.4	Versione aggiornata per la versione 2.0.4 di Greengrass nucleus.
2.0.3	Versione iniziale.

Rilevatore IP

Il componente del rilevatore IP (`aws.greengrass.clientdevices.IPDetector`) esegue le seguenti operazioni:

- Monitora le informazioni sulla connettività di rete del dispositivo principale Greengrass. Queste informazioni includono gli endpoint di rete del dispositivo principale e la porta su cui opera un broker MQTT.
- Aggiorna le informazioni di connettività del dispositivo principale nel AWS IoT Greengrass servizio cloud.

I dispositivi client possono utilizzare Greengrass cloud discovery per recuperare le informazioni di connettività dei dispositivi principali associati. Quindi, i dispositivi client possono provare a connettersi a ciascun dispositivo principale finché non si connettono correttamente.

> **Note**
>
> I dispositivi client sono dispositivi IoT locali che si connettono a un dispositivo core Greengrass per inviare messaggi MQTT e dati da elaborare. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

Il componente del rilevatore IP sostituisce le informazioni di connettività esistenti di un dispositivo principale con le informazioni rilevate. Poiché questo componente rimuove le informazioni esistenti, è possibile utilizzare il componente del rilevatore IP o gestire manualmente le informazioni di connettività.

> **Note**
>
> Il componente del rilevatore IP rileva solo gli indirizzi IPv4.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x
- 2,0x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Il ruolo del servizio Greengrass deve essere associato al tuo Account AWS e consentire le autorizzazioni `iot:GetThingShadow` `eiot:UpdateThingShadow`.
- La AWS IoT politica del dispositivo principale deve consentire l'`greengrass:UpdateConnectivityInfo`autorizzazione. Per ulteriori informazioni, consulta Policy AWS IoT per operazioni del piano dei dati e AWS IoTPolitica minima per supportare i dispositivi client.
- Se si configura il componente broker MQTT del dispositivo principale per utilizzare una porta diversa dalla porta predefinita 8883, è necessario utilizzare IP detector v2.1.0 o versione successiva. Configuralo per segnalare la porta in cui opera il broker.
- Se disponi di una configurazione di rete complessa, il componente del rilevatore IP potrebbe non essere in grado di identificare gli endpoint in cui i dispositivi client possono connettersi al dispositivo principale. Se il componente del rilevatore IP non è in grado di gestire gli endpoint, devi invece gestire manualmente gli endpoint principali del dispositivo. Ad esempio, se il dispositivo principale si trova dietro un router che gli inoltra la porta del broker MQTT, è necessario specificare l'indirizzo

IP del router come endpoint per il dispositivo principale. Per ulteriori informazioni, consulta Gestisci gli endpoint principali dei dispositivi.

- Il componente del rilevatore IP è supportato per l'esecuzione in un VPC.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.8 – 2.1.9

La tabella seguente elenca le dipendenze per le versioni 2.1.8 e 2.1.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.13.0	Flessibili

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.12.0	Flessibili

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.11.0	Flessibili

2.1.5

La tabella seguente elenca le dipendenze per la versione 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.10.0	Flessibili

2.1.4

La tabella seguente elenca le dipendenze per la versione 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.9.0	Flessibili

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.8.0	Flessibili

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.7.0	Flessibili

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.6.0	Flessibili

2.1.0 and 2.0.2

La tabella seguente elenca le dipendenze per le versioni 2.1.0 e 2.0.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.5.0	Flessibili

2.0.1

La tabella seguente elenca le dipendenze per la versione 2.0.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.4.0	Flessibili

2.0.0

La tabella seguente elenca le dipendenze per la versione 2.0.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.3.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

2.1.x

`defaultPort`

(Facoltativo) La porta del broker MQTT da segnalare quando questo componente rileva indirizzi IP. È necessario specificare questo parametro se si configura il broker MQTT per utilizzare una porta diversa dalla porta predefinita 8883.

Impostazione predefinita: `8883`

`includeIPv4LoopbackAddrs`

(Facoltativo) È possibile abilitare questa opzione per rilevare e segnalare gli indirizzi di loopback IPv4. Si tratta di indirizzi IP, ad esempio quelli in cui un dispositivo può comunicare con se stesso`localhost`. Utilizzate questa opzione in ambienti di test in cui il dispositivo principale e il dispositivo client funzionano sullo stesso sistema.

Impostazione predefinita: `false`

`includeIPv4LinkLocalAddrs`

(Facoltativo) È possibile abilitare questa opzione per rilevare e segnalare gli indirizzi locali del collegamento IPv4. Utilizzate questa opzione se la rete del dispositivo principale non dispone di indirizzi IP assegnati staticamente o del Dynamic Host Configuration Protocol (DHCP).

Impostazione predefinita: `false`

2.0.x

`includeIPv4LoopbackAddrs`

(Facoltativo) È possibile abilitare questa opzione per rilevare e segnalare gli indirizzi di loopback IPv4. Si tratta di indirizzi IP, ad esempio quelli in cui un dispositivo può comunicare con se stesso`localhost`. Utilizzate questa opzione in ambienti di test in cui il dispositivo principale e il dispositivo client funzionano sullo stesso sistema.

Impostazione predefinita: `false`

`includeIPv4LinkLocalAddrs`

(Facoltativo) È possibile abilitare questa opzione per rilevare e segnalare gli indirizzi locali del collegamento IPv4. Utilizzate questa opzione se la rete del dispositivo principale non dispone di indirizzi IP assegnati staticamente o del Dynamic Host Configuration Protocol (DHCP).

Impostazione predefinita: `false`

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.1.9	Correzioni di bug e miglioramenti • Regola la fase di acquisizione dell'IP in modo da inviare i log solo a livello di registro di debug.

Versione	Modifiche
2.1.8	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.1.7	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.5	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.4	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.2	Correzioni di bug e miglioramenti • Migliora i messaggi di errore registrati da questo componente in determinati scenari. • Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.1	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.1.0	Miglioramenti • Aggiunge il `defaultPort` parametro, che consente di utilizzare una porta broker MQTT non predefinita. • Aggiornamenti per rendere più chiari i messaggi di registro.
2.0.2	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.1	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.0.0	Versione iniziale.

Firehose

Il componente Firehose (`aws.greengrass.KinesisFirehose`) pubblica i dati tramite i flussi di distribuzione di Amazon Data Firehose verso destinazioni come Amazon S3, Amazon Redshift e Amazon Service. OpenSearch Per ulteriori informazioni, consulta What is Amazon Data Firehose? nella Amazon Data Firehose Developer Guide.

Per pubblicare su un flusso di distribuzione Kinesis con questo componente, pubblica un messaggio su un argomento a cui questo componente è abbonato. Per impostazione predefinita, questo componente sottoscrive gli argomenti di `kinesisfirehose/message` pubblicazione/sottoscrizione `kinesisfirehose/message/binary/#` locali. È possibile specificare altri argomenti, inclusi gli argomenti AWS IoT Core MQTT, quando si distribuisce questo componente.

> **Note**
>
> Questo componente offre funzionalità simili al connettore Firehose della versione 1. AWS IoT Greengrass Per ulteriori informazioni, consulta il connettore Firehose nella AWS IoT Greengrass V1 Developer Guide.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x
- 2,0x

Type

Questo componente è un componente Lambda () `aws.greengrass.lambda`. Il nucleo Greengrass esegue la funzione Lambda di questo componente utilizzando il componente di avvio Lambda.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato solo sui dispositivi principali di Linux.

Requisiti

Questo componente ha i seguenti requisiti:

- Il dispositivo principale deve soddisfare i requisiti per eseguire le funzioni Lambda. Se desideri che il dispositivo principale esegua funzioni Lambda containerizzate, il dispositivo deve soddisfare i requisiti per farlo. Per ulteriori informazioni, consulta Requisiti della funzione Lambda.
- Python versione 3.7 installata sul dispositivo principale e aggiunta alla variabile di ambiente PATH.
- Il ruolo del dispositivo Greengrass deve consentire le `firehose:PutRecordBatch` azioni `firehose:PutRecord` and, come illustrato nel seguente esempio di policy IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:firehose:region:account-id:deliverystream/stream-name"
      ]
    }
  ]
}
```

È possibile sovrascrivere dinamicamente il flusso di consegna predefinito nel payload del messaggio di input per questo componente. Se l'applicazione utilizza questa funzionalità, la policy

IAM deve includere tutti i flussi di destinazione come risorse. Puoi concedere alle risorse un accesso granulare o condizionale (ad esempio, utilizzando uno schema di denominazione con il carattere jolly *).

- Per ricevere i dati di output da questo componente, è necessario unire il seguente aggiornamento di configurazione per il componente legacy del router di abbonamento (`aws.greengrass.LegacySubscriptionRouter`) quando si distribuisce questo componente. Questa configurazione specifica l'argomento in cui questo componente pubblica le risposte.

  Legacy subscription router v2.1.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-kinesisfirehose": {
        "id": "aws-greengrass-kinesisfirehose",
        "source": "component:aws.greengrass.KinesisFirehose",
        "subject": "kinesisfirehose/message/status",
        "target": "cloud"
      }
    }
  }
  ```

  Legacy subscription router v2.0.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-kinesisfirehose": {
        "id": "aws-greengrass-kinesisfirehose",
        "source": "arn:aws:lambda:region:aws:function:aws-greengrass-
  kinesisfirehose:version",
        "subject": "kinesisfirehose/message/status",
        "target": "cloud"
      }
    }
  }
  ```

  - Sostituisci la *`regione`* con quella Regione AWS che usi.
  - Sostituisci la *`versione`* con la versione della funzione Lambda eseguita da questo componente. Per trovare la versione della funzione Lambda, è necessario visualizzare la ricetta per la versione di questo componente che si desidera distribuire. Apri la pagina dei dettagli di questo componente nella AWS IoT Greengrass console e cerca la coppia chiave-

valore della funzione Lambda. Questa coppia chiave-valore contiene il nome e la versione della funzione Lambda.

> **Important**
>
> È necessario aggiornare la versione della funzione Lambda sul router di abbonamento legacy ogni volta che si distribuisce questo componente. Ciò garantisce l'utilizzo della versione corretta della funzione Lambda per la versione del componente che si distribuisce.

Per ulteriori informazioni, consulta Creare distribuzione.

- Il componente Firehose è supportato per l'esecuzione in un VPC. Per distribuire questo componente in un VPC, è necessario quanto segue.
  - Il componente Firehose deve disporre di una connettività a `firehose.region.amazonaws.com` cui l'endpoint VPC sia di. `com.amazonaws.region.kinesis-firehose`

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`firehose.` *`region`*`.amazonaws.com`	443	Sì	Caricare i dati su Firehose.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le

dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.5

La tabella seguente elenca le dipendenze per la versione 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.4

La tabella seguente elenca le dipendenze per la versione 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.8 - 2.1.0

La tabella seguente elenca le dipendenze per le versioni 2.0.8 e 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili

Dipendenza	Versioni compatibili	Tipo di dipendenza
Servizio di scambio di token	^2.0.0	Rigidi

2.0.7

La tabella seguente elenca le dipendenze per la versione 2.0.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.6

La tabella seguente elenca le dipendenze per la versione 2.0.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.5

La tabella seguente elenca le dipendenze per la versione 2.0.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.4

La tabella seguente elenca le dipendenze per la versione 2.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.3

La tabella seguente elenca le dipendenze per la versione 2.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Rigidi
Lanciatore Lambda	>=1.0.0	Rigidi
Runtime Lambda	>=1.0.0	Flessibili
Servizio di scambio di token	>=1.0.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

> **Note**
>
> La configurazione predefinita di questo componente include i parametri della funzione Lambda. Ti consigliamo di modificare solo i seguenti parametri per configurare questo componente sui tuoi dispositivi.

`lambdaParams`

Un oggetto che contiene i parametri per la funzione Lambda di questo componente. Questo oggetto contiene le seguenti informazioni:

`EnvironmentVariables`

Un oggetto che contiene i parametri della funzione Lambda. Questo oggetto contiene le seguenti informazioni:

`DEFAULT_DELIVERY_STREAM_ARN`

L'ARN del flusso di distribuzione Firehose predefinito a cui il componente invia i dati. È possibile sovrascrivere lo stream di destinazione con la `delivery_stream_arn` proprietà nel payload del messaggio di input.

> **Note**
>
> Il ruolo principale del dispositivo deve consentire le azioni richieste su tutti i flussi di consegna di destinazione. Per ulteriori informazioni, consulta Requisiti.

`PUBLISH_INTERVAL`

(Facoltativo) Il numero massimo di secondi di attesa prima che il componente pubblichi dati in batch su Firehose. Per configurare il componente in modo che pubblichi le metriche man mano che le riceve, vale a dire senza raggruppamento in batch, specificate. `0`

Questo valore può essere al massimo di 900 secondi.

Impostazione predefinita: 10 secondi

`DELIVERY_STREAM_QUEUE_SIZE`

(Facoltativo) Il numero massimo di record da conservare in memoria prima che il componente rifiuti nuovi record per lo stesso flusso di distribuzione.

Questo valore deve essere di almeno 2.000 record.

Impostazione predefinita: 5.000 record

`containerMode`

(Facoltativo) La modalità di containerizzazione per questo componente. Seleziona una delle opzioni seguenti:

- `NoContainer`— Il componente non viene eseguito in un ambiente di runtime isolato.
- `GreengrassContainer`— Il componente viene eseguito in un ambiente di runtime isolato all'interno del AWS IoT Greengrass contenitore.

Impostazione predefinita: `GreengrassContainer`

`containerParams`

(Facoltativo) Un oggetto che contiene i parametri del contenitore per questo componente. Il componente utilizza questi parametri se si specifica `GreengrassContainer` per`containerMode`.

Questo oggetto contiene le seguenti informazioni:

`memorySize`

(Facoltativo) La quantità di memoria (in kilobyte) da allocare al componente.

Il valore predefinito è 64 MB (65.535 KB).

`pubsubTopics`

(Facoltativo) Un oggetto che contiene gli argomenti a cui il componente sottoscrive la sottoscrizione per ricevere messaggi. È possibile specificare ogni argomento e se il componente sottoscrive gli argomenti MQTT AWS IoT Core o gli argomenti di pubblicazione/sottoscrizione locali.

Questo oggetto contiene le seguenti informazioni:

`0`— Si tratta di un indice di matrice sotto forma di stringa.

Un oggetto che contiene le seguenti informazioni:

`type`

(Facoltativo) Il tipo di messaggi di pubblicazione/sottoscrizione utilizzato da questo componente per sottoscrivere i messaggi. Seleziona una delle opzioni seguenti:

- `PUB_SUB`: iscriviti ai messaggi di pubblicazione/sottoscrizione locali. Se scegli questa opzione, l'argomento non può contenere caratteri jolly MQTT. Per ulteriori informazioni su come inviare messaggi dal componente personalizzato quando si specifica questa opzione, vedere. Pubblicare/sottoscrivere messaggi locali
- `IOT_CORE`— Abbonarsi ai messaggi AWS IoT Core MQTT. Se scegli questa opzione, l'argomento può contenere caratteri jolly MQTT. Per ulteriori informazioni su come inviare messaggi da componenti personalizzati quando si specifica questa opzione, vedere. AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

Impostazione predefinita: `PUB_SUB`

`topic`

(Facoltativo) L'argomento a cui il componente si iscrive per ricevere messaggi. Se si specifica `IotCore` for`type`, è possibile utilizzare i caratteri jolly MQTT (+and#) in questo argomento.

Example Esempio: aggiornamento basato sull'unione della configurazione (modalità contenitore)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_DELIVERY_STREAM_ARN": "arn:aws:firehose:us-
west-2:123456789012:deliverystream/mystream"
    }
  },
  "containerMode": "GreengrassContainer"
}
```

Example Esempio: aggiornamento tramite fusione della configurazione (nessuna modalità contenitore)

```
{
```

```
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_DELIVERY_STREAM_ARN": "arn:aws:firehose:us-
west-2:123456789012:deliverystream/mystream"
    }
  },
  "containerMode": "NoContainer"
}
```

Dati di input

Questo componente accetta contenuti in streaming sui seguenti argomenti e li invia al flusso di consegna di destinazione. Il componente accetta due tipi di dati di input:

- Dati JSON nell'argomento `kinesisfirehose/message`.
- Dati binari nell'argomento `kinesisfirehose/message/binary/#`.

Argomento predefinito per i dati JSON (pubblicazione locale/sottoscrizione): `kinesisfirehose/message`

Il messaggio accetta le seguenti proprietà. I messaggi di input devono essere in formato JSON.

`request`

I dati da inviare al flusso di distribuzione e al flusso di distribuzione di destinazione, se diverso da quello predefinito.

Tipo: `object` che contiene le seguenti informazioni:

`data`

I dati da inviare al flusso di distribuzione.

Tipo: `string`

`delivery_stream_arn`

(Facoltativo) L'ARN del flusso di distribuzione Firehose di destinazione. Specificate questa proprietà per sovrascrivere il flusso di consegna predefinito.

Tipo: `string`

`id`

Un ID arbitrario della richiesta. Utilizzate questa proprietà per mappare una richiesta di input a una risposta di output. Quando specificate questa proprietà, il componente imposta la `id` proprietà nell'oggetto di risposta su questo valore.

Tipo: `string`

Example Input di esempio

```
{
  "request": {
    "delivery_stream_arn": "arn:aws:firehose:region:account-id:deliverystream/
stream2-name",
    "data": "Data to send to the delivery stream."
  },
  "id": "request123"
}
```

Argomento predefinito per i dati binari (pubblicazione locale/sottoscrizione): `kinesisfirehose/message/binary/#`

Utilizzare questo argomento per inviare un messaggio contenente dati binari. Il componente non analizza i dati binari. Il componente trasmette i dati così come sono.

Per associare la richiesta di input a una risposta di output, sostituisci il carattere jolly # nell'argomento del messaggio con un ID richiesta arbitrario. Ad esempio, se pubblichi un messaggio in `kinesisfirehose/message/binary/request123`, la proprietà `id` nell'oggetto di risposta viene impostata su `request123`.

Se non desideri associare una richiesta a una risposta, puoi pubblicare i messaggi in `kinesisfirehose/message/binary/`. Assicurati di includere la barra finale (). `/`

Dati di output

Per impostazione predefinita, questo componente pubblica le risposte come dati di output sul seguente argomento MQTT. È necessario specificare questo argomento come contenuto `subject` nella configurazione del componente legacy del router di abbonamento. Per ulteriori informazioni su come sottoscrivere i messaggi relativi a questo argomento nei componenti personalizzati, consultaAWS IoT Core Pubblicare/sottoscrivere messaggi MQTT.

Argomento predefinito (AWS IoT Core MQTT): `kinesisfirehose/message/status`

Example Output di esempio

La risposta contiene lo stato di ogni record di dati inviati nel batch.

```
{
  "response": [
    {
      "ErrorCode": "error",
      "ErrorMessage": "test error",
      "id": "request123",
      "status": "fail"
    },
    {
      "firehose_record_id": "xyz2",
      "id": "request456",
      "status": "success"
    },
    {
      "firehose_record_id": "xyz3",
      "id": "request890",
      "status": "success"
    }
  ]
}
```

> **Note**
>
> Se il componente rileva un errore che può essere riprovato, ad esempio un errore di connessione, riprova la pubblicazione nel batch successivo.

File di registro locale

Questo componente utilizza il seguente file di registro.

```
/greengrass/v2/logs/aws.greengrass.KinesisFirehose.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale.

```
sudo tail -f /greengrass/v2/logs/aws.greengrass.KinesisFirehose.log
```

Licenze

Questo componente include i seguenti software/licenze di terze parti:

- AWS SDK for Python (Boto3)/Apache License 2.0
- botocore/Apache License 2.0
- dateutil/PSF License
- docutils/BSD License, GNU General Public License (GPL), Python Software Foundation License, Public Domain
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/MIT License

Questo componente è rilasciato in base al contratto di licenza del software Greengrass Core.

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
2.1.7	Versione aggiornata per Greengrass nucleus versione 2.12.0.
2.1.6	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.5	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.4	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.

Versione	Modifiche
2.1.3	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per Greengrass nucleus versione 2.7.0.
2.1.1	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.0	Nuove funzionalità • Aggiunge il supporto per le configurazioni proxy di rete HTTPS. Per ulteriori informazioni, consultare Connessione alla porta 443 o tramite un proxy di rete e Abilita il dispositivo principale in modo che consideri attendibile un proxy HTTPS.
2.0.8	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.0.7	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.6	Versione aggiornata per Greengrass nucleus versione 2.3.0.
2.0.5	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.0.4	Versione aggiornata per la versione 2.1.0 di Greengrass nucleus.
2.0.3	Versione iniziale.

Consulta anche

- Che cos'è Amazon Data Firehose? nella Guida per sviluppatori di Amazon Data Firehose

Lanciatore Lambda

Il componente di avvio Lambda (`aws.greengrass.LambdaLauncher`) avvia e interrompe AWS Lambda le funzioni sui AWS IoT Greengrass dispositivi principali. Questo componente imposta anche qualsiasi containerizzazione ed esegue i processi secondo gli utenti specificati.

> **Note**
>
> Quando si distribuisce un componente della funzione Lambda su un dispositivo principale, la distribuzione include anche questo componente. Per ulteriori informazioni, consulta Esegui AWS Lambda funzioni.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.0.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato solo sui dispositivi principali di Linux.

Requisiti

Questo componente ha i seguenti requisiti:

- Il dispositivo principale deve soddisfare i requisiti per eseguire le funzioni Lambda. Se desideri che il dispositivo principale esegua funzioni Lambda containerizzate, il dispositivo deve soddisfare i requisiti per farlo. Per ulteriori informazioni, consulta Requisiti della funzione Lambda.
- Il componente di avvio Lambda è supportato per l'esecuzione in un VPC.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.0.11 – 2.0.13

La tabella seguente elenca le dipendenze per le versioni da 2.0.11 a 2.0.13 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Gestore Lambda	>=2.0.0 <2.4.0	Rigidi

2.0.9 – 2.0.10

La tabella seguente elenca le dipendenze per le versioni da 2.0.9 a 2.0.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Gestore Lambda	>=2.0.0 <2.3.0	Rigidi

2.0.4 - 2.0.8

La tabella seguente elenca le dipendenze per le versioni da 2.0.4 a 2.0.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Gestore Lambda	>=2.0.0 <2.2.0	Rigidi

2.0.3

La tabella seguente elenca le dipendenze per la versione 2.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Gestore Lambda	>=2.0.3 <2.1.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente non ha parametri di configurazione.

File di registro locale

Questo componente utilizza il seguente file di registro.

```
/greengrass/v2/logs/lambdaFunctionComponentName.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale e sostituisci *lambdaFunctionComponentName* con il nome del componente della funzione Lambda avviato da questo componente.

  ```
  sudo tail -f /greengrass/v2/logs/lambdaFunctionComponentName.log
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.0.13	Correzioni di bug e miglioramenti Correzioni di bug generali e miglioramenti.
2.0.12	Correzioni di bug e miglioramenti Risolve un problema per cui il programma di avvio Lambda poteva generare un errore se il processo precedente non veniva interrotto correttamente.
2.0.11	Support per Lambda manager 2.3.0.
2.0.10	Correzioni di bug e miglioramenti • Correzioni di bug generali e miglioramenti.
2.0.9	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.0.8	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.7	Versione aggiornata per Greengrass nucleus versione 2.3.0.
2.0.6	Miglioramenti delle prestazioni generali e correzioni di bug.
2.0.4	Correzioni di bug e miglioramenti • Risolve un problema per cui il componente non passa correttamente `AddGroupOwner` al contenitore delle funzioni Lambda.
2.0.3	Versione iniziale.

Gestore Lambda

Il componente Lambda manager (`aws.greengrass.LambdaManager`) gestisce gli elementi di lavoro e la comunicazione tra processi per AWS Lambda le funzioni eseguite sul dispositivo principale Greengrass.

> **Note**
>
> Quando si distribuisce un componente della funzione Lambda su un dispositivo principale, la distribuzione include anche questo componente. Per ulteriori informazioni, consulta Esegui AWS Lambda funzioni.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.3.x
- 2.2.x
- 2.1.x
- 2,0x

Sistema operativo

Questo componente può essere installato solo su dispositivi core Linux.

Type

Questo componente è un componente del plugin (`aws.greengrass.plugin`). Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Requisiti

Questo componente ha i seguenti requisiti:

- Il dispositivo principale deve soddisfare i requisiti per eseguire le funzioni Lambda. Se desideri che il dispositivo principale esegua funzioni Lambda containerizzate, il dispositivo deve soddisfare i requisiti per farlo. Per ulteriori informazioni, consulta Requisiti della funzione Lambda.
- Il componente Lambda manager è supportato per l'esecuzione in un VPC.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.3.2 and 2.3.3

La tabella seguente elenca le dipendenze per le versioni 2.3.2 e 2.3.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili

2.2.10 and 2.3.1

La tabella seguente elenca le dipendenze per le versioni 2.2.10 e 2.3.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili

2.2.8 and 2.2.9

La tabella seguente elenca le dipendenze per le versioni 2.2.8 e 2.2.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili

2.2.7

La tabella seguente elenca le dipendenze per la versione 2.2.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili

2.2.6

La tabella seguente elenca le dipendenze per la versione 2.2.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili

2.2.5

La tabella seguente elenca le dipendenze per la versione 2.2.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili

2.2.4

La tabella seguente elenca le dipendenze per la versione 2.2.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili

2.2.1 - 2.2.3

La tabella seguente elenca le dipendenze per le versioni da 2.2.1 a 2.2.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili

2.2.0

La tabella seguente elenca le dipendenze per la versione 2.2.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.6.0	Flessibili

2.1.3 and 2.1.4

La tabella seguente elenca le dipendenze per le versioni 2.1.3 e 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili

2.0.x

La tabella seguente elenca le dipendenze per la versione 2.0.x di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`logHandlerMode`

> **Note**
>
> Solo per le versioni 2.3.0+ di lambda manager

Utilizzato per scegliere l'implementazione del Lambda log manager da utilizzare. Imposta il valore in modo `optimized` da utilizzare meno thread per leggere i log lambda.

`getResultTimeoutInSecond`

(Facoltativo) La quantità massima di tempo in secondi in cui le funzioni Lambda possono essere eseguite prima del timeout.

Impostazione predefinita: `60`

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

```
/greengrass/v2/logs/greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale.

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
2.3.3	Correzioni di bug e miglioramenti • Correzioni di bug generali e miglioramenti.
2.3.2	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.3.1	Correzioni di bug e miglioramenti • Regola i livelli di registro per determinati errori.
2.3.0	Nuove funzionalità • Il gestore di log è stato ottimizzato per ridurre il carico della CPU. Utilizzate questa funzionalità impostando l'opzione di configurazione `logHandlerMode suoptimized` . Correzioni di bug e miglioramenti • Non registra più l'intero stacktrace`WorkQueueFullException` , migliorando i log e le prestazioni. • Imposta il timeout di spegnimento lambda da 15 secondi a 300 secondi per evitare i timeout di spegnimento. • Risolve un problema per cui le lambda su richiesta potrebbero non riuscire a riavviarsi dopo aver modificato la configurazione.
2.2.11	Correzioni di bug e miglioramenti • Risolve un problema per cui la LegacySubscriptionRouter configura zione non si aggiorna quando la configurazione Lambda cambia.
2.2.10	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.2.9	Correzioni di bug e miglioramenti Risolve un problema per cui il numero di porta veniva danneggiato a causa di un orologio distorto.
2.2.8	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.2.7	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.

Versione	Modifiche
2.2.6	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.2.5	Nuove funzionalità • Aggiunge il supporto per i caratteri jolly degli argomenti MQTT nelle fonti di eventi in cui ci si iscrive ai messaggi di pubblicazione/sottoscrizione locali. Questa funzionalità richiede la versione 2.6.0 o successiva del componente Greengrass nucleus. • Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.2.4	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.2.3	Correzioni di bug e miglioramenti • Risolve un problema per cui più istanze di una funzione Lambda condividono un singolo cgroup. Questo componente utilizza cgroups per gestire l'utilizzo delle risorse per le funzioni Lambda.
2.2.2	Correzioni di bug e miglioramenti • Risolve un problema a causa del quale i componenti della funzione Lambda bloccati si riavviano in modo imprevisto in determinati scenari.
2.2.1	Correzioni di bug e miglioramenti • Modifica i vincoli di versione della dipendenza Greengrass nucleus di questo componente per risolvere un problema di risoluzione delle dipendenze.

Versione	Modifiche
2.2.0	Correzioni di bug e miglioramenti • Risolve un problema per cui le funzioni Lambda non potevano scrivere i log dopo un riavvio. • Risolve un problema a causa del quale il router di sottoscrizione legacy invia messaggi duplicati quando nell'argomento sono presenti caratteri jolly. • Risolve un problema a causa del quale le funzioni Lambda non bloccate non potevano utilizzare la libreria di comunicazione interprocesso (IPC) Greengrass in. SDK per dispositivi AWS IoT
2.1.4	Correzioni di bug e miglioramenti • Risolve un problema che causava l'elaborazione di un solo messaggio da parte delle funzioni Lambda che utilizzano i runtime NodeJS. • Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.1.1	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.1.0	Versione aggiornata per la versione 2.1.0 di Greengrass nucleus.
2.0.3	Versione iniziale.

Runtime Lambda

Il componente Lambda runtimes (`aws.greengrass.LambdaRuntimes`) fornisce i runtime utilizzati dai dispositivi core Greengrass per eseguire le funzioni. AWS Lambda

> **Note**
>
> Quando si distribuisce un componente della funzione Lambda su un dispositivo principale, la distribuzione include anche questo componente. Per ulteriori informazioni, consulta Esegui AWS Lambda funzioni.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.0.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato solo sui dispositivi principali di Linux.

Requisiti

Questo componente ha i seguenti requisiti:

- Il dispositivo principale deve soddisfare i requisiti per eseguire le funzioni Lambda. Se desideri che il dispositivo principale esegua funzioni Lambda containerizzate, il dispositivo deve soddisfare i requisiti per farlo. Per ulteriori informazioni, consulta Requisiti della funzione Lambda.
- Il componente Lambda runtimes è supportato per l'esecuzione in un VPC.

Dipendenze

Questo componente non ha alcuna dipendenza.

Configurazione

Questo componente non ha parametri di configurazione.

File di registro locale

Questo componente non emette registri.

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.0.8	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.0.7	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.6	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.0.5	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.0.4	Versione aggiornata per la versione 2.1.0 di Greengrass nucleus.
2.0.3	Versione iniziale.

Router di abbonamento legacy

Il router di abbonamento legacy (`aws.greengrass.LegacySubscriptionRouter`) gestisce gli abbonamenti sul dispositivo principale Greengrass. Gli abbonamenti sono una funzionalità della AWS IoT Greengrass V1 che definisce gli argomenti che le funzioni Lambda possono utilizzare per la messaggistica MQTT su un dispositivo principale. Per ulteriori informazioni, consulta Sottoscrizioni gestite nel flusso di lavoro di messaggistica MQTT nella V1 Developer Guide. AWS IoT Greengrass

Puoi utilizzare questo componente per abilitare gli abbonamenti per i componenti del connettore e i componenti della funzione Lambda che utilizzano Core SDKAWS IoT Greengrass.

> **Note**
>
> Il componente legacy del router di abbonamento è richiesto solo se la funzione Lambda utilizza la `publish()` funzione nel AWS IoT Greengrass Core SDK. Se aggiorni il codice della funzione Lambda per utilizzare l'interfaccia di comunicazione tra processi (IPC) nella SDK per dispositivi AWS IoT V2, non è necessario implementare il componente legacy del router di abbonamento. Per ulteriori informazioni, consulta i seguenti servizi di comunicazione tra processi:
>
> - Pubblicare/sottoscrivere messaggi locali
> - AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x
- 2,0x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato solo sui dispositivi principali di Linux.

Requisiti

Questo componente ha i seguenti requisiti:

- Il router di abbonamento legacy è supportato per l'esecuzione in un VPC.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.11

La tabella seguente elenca le dipendenze per la versione 2.1.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili

2.1.10

La tabella seguente elenca le dipendenze per la versione 2.1.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili

2.1.9

La tabella seguente elenca le dipendenze per la versione 2.1.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili

2.1.8

La tabella seguente elenca le dipendenze per la versione 2.1.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili

2.1.5

La tabella seguente elenca le dipendenze per la versione 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili

2.1.4

La tabella seguente elenca le dipendenze per la versione 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili

2.0.3

La tabella seguente elenca le dipendenze per la versione 2.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

v2.1.x

`subscriptions`

(Facoltativo) Gli abbonamenti da abilitare sul dispositivo principale. Si tratta di un oggetto, in cui ogni chiave è un ID univoco e ogni valore è un oggetto che definisce l'abbonamento per quel connettore. È necessario configurare un abbonamento quando si distribuisce un componente connettore V1 o una funzione Lambda che utilizza Core SDK. AWS IoT Greengrass

Ogni oggetto di sottoscrizione contiene le seguenti informazioni:

`id`

L'ID univoco di questo abbonamento. Questo ID deve corrispondere alla chiave per questo oggetto di sottoscrizione.

`source`

La funzione Lambda che utilizza AWS IoT Greengrass Core SDK per pubblicare messaggi MQTT sugli argomenti specificati. `subject` Specifica una delle seguenti proprietà:

- Il nome di un componente della funzione Lambda sul dispositivo principale. Specificate il nome del componente con il `component:` prefisso, ad esempio. **`component:com.example.HelloWorldLambda`**
- L'Amazon Resource Name (ARN) di una funzione Lambda sul dispositivo principale.

> **Important**
>
> Se la versione della funzione Lambda cambia, è necessario configurare l'abbonamento con la nuova versione della funzione. Altrimenti, questo componente non indirizzerà i messaggi finché la versione non corrisponderà all'abbonamento.
> È necessario specificare un Amazon Resource Name (ARN) che includa la versione della funzione da importare. Non è possibile utilizzare alias di versione come `$LATEST`.

Per distribuire un abbonamento per un componente del connettore V1, specifica il nome del componente o l'ARN della funzione Lambda del componente connettore.

`subject`

L'argomento o il filtro degli argomenti MQTT su cui l'origine e la destinazione possono pubblicare e ricevere messaggi. Questo valore supporta i caratteri jolly degli # argomenti + e degli argomenti.

`target`

La destinazione che riceve i messaggi MQTT sugli argomenti specificati in. `subject` L'abbonamento specifica che la `source` funzione pubblica messaggi MQTT su AWS IoT Core o verso una funzione Lambda sul dispositivo principale. Specifica una delle seguenti proprietà:

- `cloud`. La `source` funzione pubblica messaggi MQTT su. AWS IoT Core
- Il nome di un componente della funzione Lambda sul dispositivo principale. Specificate il nome del componente con il `component:` prefisso, ad esempio. **`component:com.example.HelloWorldLambda`**

- L'Amazon Resource Name (ARN) di una funzione Lambda sul dispositivo principale.

> **Important**
>
> Se la versione della funzione Lambda cambia, è necessario configurare l'abbonamento con la nuova versione della funzione. Altrimenti, questo componente non indirizzerà i messaggi finché la versione non corrisponderà all'abbonamento.
> È necessario specificare un Amazon Resource Name (ARN) che includa la versione della funzione da importare. Non è possibile utilizzare alias di versione come `$LATEST`.

Impostazione predefinita: nessun abbonamento

Example Esempio di aggiornamento della configurazione (definizione di un abbonamento aAWS IoT Core)

L'esempio seguente specifica che il componente della funzione `com.example.HelloWorldLambda` Lambda pubblica un messaggio AWS IoT Core MQTT sull'argomento. `hello/world`

```
{
  "subscriptions": {
    "Greengrass_HelloWorld_to_cloud": {
      "id": "Greengrass_HelloWorld_to_cloud",
      "source": "component:com.example.HelloWorldLambda",
      "subject": "hello/world",
      "target": "cloud"
    }
  }
}
```

Example Esempio di aggiornamento della configurazione (definizione di un abbonamento a un'altra funzione Lambda)

L'esempio seguente specifica che il componente funzione `com.example.HelloWorldLambda` Lambda pubblica messaggi MQTT sul componente della funzione `com.example.MessageRelay` Lambda sull'argomento. `hello/world`

```
{
  "subscriptions": {
    "Greengrass_HelloWorld_to_MessageRelay": {
      "id": "Greengrass_HelloWorld_to_MessageRelay",
      "source": "component:com.example.HelloWorldLambda",
      "subject": "hello/world",
      "target": "component:com.example.MessageRelay"
    }
  }
}
```

v2.0.x

`subscriptions`

(Facoltativo) Gli abbonamenti da abilitare sul dispositivo principale. Si tratta di un oggetto, in cui ogni chiave è un ID univoco e ogni valore è un oggetto che definisce l'abbonamento per quel connettore. È necessario configurare un abbonamento quando si distribuisce un componente connettore V1 o una funzione Lambda che utilizza Core SDK. AWS IoT Greengrass

Ogni oggetto di sottoscrizione contiene le seguenti informazioni:

`id`

L'ID univoco di questo abbonamento. Questo ID deve corrispondere alla chiave per questo oggetto di sottoscrizione.

`source`

La funzione Lambda che utilizza AWS IoT Greengrass Core SDK per pubblicare messaggi MQTT sugli argomenti specificati. `subject` Specificare le impostazioni seguenti:

- L'Amazon Resource Name (ARN) di una funzione Lambda sul dispositivo principale.

> **Important**
>
> Se la versione della funzione Lambda cambia, è necessario configurare l'abbonamento con la nuova versione della funzione. Altrimenti, questo componente non indirizzerà i messaggi finché la versione non corrisponderà all'abbonamento.

È necessario specificare un Amazon Resource Name (ARN) che includa la versione della funzione da importare. Non è possibile utilizzare alias di versione come `$LATEST`.

Per distribuire un abbonamento per un componente del connettore V1, specifica l'ARN della funzione Lambda del componente connettore.

`subject`

L'argomento o il filtro degli argomenti MQTT su cui l'origine e la destinazione possono pubblicare e ricevere messaggi. Questo valore supporta i caratteri jolly degli # argomenti + e degli argomenti.

`target`

La destinazione che riceve i messaggi MQTT sugli argomenti specificati in. `subject` L'abbonamento specifica che la `source` funzione pubblica messaggi MQTT su AWS IoT Core o verso una funzione Lambda sul dispositivo principale. Specifica una delle seguenti proprietà:

- `cloud`. La `source` funzione pubblica messaggi MQTT su. AWS IoT Core
- L'Amazon Resource Name (ARN) di una funzione Lambda sul dispositivo principale.

> **Important**
>
> Se la versione della funzione Lambda cambia, è necessario configurare l'abbonamento con la nuova versione della funzione. Altrimenti, questo componente non indirizzerà i messaggi finché la versione non corrisponderà all'abbonamento.
> È necessario specificare un Amazon Resource Name (ARN) che includa la versione della funzione da importare. Non è possibile utilizzare alias di versione come `$LATEST`.

Impostazione predefinita: nessun abbonamento

Example Esempio di aggiornamento della configurazione (definizione di un abbonamento aAWS IoT Core)

L'esempio seguente specifica che la `Greengrass_HelloWorld` funzione pubblica un messaggio MQTT AWS IoT Core sull'argomento. `hello/world`

```
"subscriptions": {
  "Greengrass_HelloWorld_to_cloud": {
    "id": "Greengrass_HelloWorld_to_cloud",
    "source": "arn:aws:lambda:us-
west-2:123456789012:function:Greengrass_HelloWorld:5",
    "subject": "hello/world",
    "target": "cloud"
  }
}
```

Example Esempio di aggiornamento della configurazione (definizione di un abbonamento a un'altra funzione Lambda)

L'esempio seguente specifica che la `Greengrass_HelloWorld` funzione pubblica messaggi MQTT su on the `Greengrass_MessageRelay` topic. `hello/world`

```
"subscriptions": {
  "Greengrass_HelloWorld_to_MessageRelay": {
    "id": "Greengrass_HelloWorld_to_MessageRelay",
    "source": "arn:aws:lambda:us-
west-2:123456789012:function:Greengrass_HelloWorld:5",
    "subject": "hello/world",
    "target": "arn:aws:lambda:us-
west-2:123456789012:function:Greengrass_MessageRelay:5"
  }
}
```

File di registro locale

Questo componente non emette registri.

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.1.11	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.1.10	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.

Versione	Modifiche
2.1.9	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.8	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.7	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.5	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.4	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.1.1	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.1.0	Correzioni di bug e miglioramenti • Aggiunge il supporto per specificare i nomi dei componenti anziché gli ARN per `source` e. `target` Se si specifica il nome di un componente per un abbonamento, non è necessario riconfigurare l'abbonamento ogni volta che cambia la versione della funzione Lambda.
2.0.3	Versione iniziale.

Console di debug locale

Il componente della console di debug locale (`aws.greengrass.LocalDebugConsole`) fornisce una dashboard locale che mostra informazioni sui dispositivi AWS IoT Greengrass principali e sui relativi componenti. È possibile utilizzare questa dashboard per eseguire il debug del dispositivo principale e gestire i componenti locali.

> **Important**
>
> Ti consigliamo di utilizzare questo componente solo in ambienti di sviluppo, non in ambienti di produzione. Questo componente fornisce l'accesso a informazioni e operazioni che in genere

non sono necessarie in un ambiente di produzione. Segui il principio del privilegio minimo distribuendo questo componente solo sui dispositivi principali dove ne hai bisogno.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.3.x
- 2.2.x
- 2.1.x
- 2,0x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Utilizza un nome utente e una password per accedere alla dashboard. Il nome utente, che è`debug`, viene fornito all'utente. È necessario utilizzare la AWS IoT Greengrass CLI per creare una password temporanea che ti autentichi con la dashboard su un dispositivo principale. È necessario essere in grado di utilizzare la AWS IoT Greengrass CLI per utilizzare la console di debug locale. Per ulteriori informazioni, consulta i requisiti della CLI di Greengrass. Per ulteriori informazioni su come generare la password e accedere, vedere Utilizzo dei componenti della console di debug locale.
- Il componente della console di debug locale è supportato per l'esecuzione in un VPC.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.4.1 – 2.4.2

La tabella seguente elenca le dipendenze per le versioni da 2.4.1 a 2.4.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.10.0 <2.13.0	Rigidi

Dipendenza	Versioni compatibili	Tipo di dipendenza
Greengrass CLI	>=2.10.0 <2.13.0	Rigidi

2.4.0

La tabella seguente elenca le dipendenze per la versione 2.4.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.10.0 <2.12.0	Rigidi
Greengrass CLI	>=2.10.0 <2.12.0	Rigidi

2.3.0 and 2.3.1

La tabella seguente elenca le dipendenze per le versioni 2.3.0 e 2.3.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.10.0 <2.12.0	Rigidi
Greengrass CLI	>=2.10.0 <2.12.0	Rigidi

2.2.9

La tabella seguente elenca le dipendenze per la versione 2.2.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.12.0	Rigidi
Greengrass CLI	>=2.1.0 <2.12.0	Rigidi

2.2.8

La tabella seguente elenca le dipendenze per la versione 2.2.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.11.0	Rigidi
Greengrass CLI	>=2.1.0 <2.11.0	Rigidi

2.2.7

La tabella seguente elenca le dipendenze per la versione 2.2.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.10.0	Rigidi
Greengrass CLI	>=2.1.0 <2.10.0	Rigidi

2.2.6

La tabella seguente elenca le dipendenze per la versione 2.2.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.9.0	Rigidi
Greengrass CLI	>=2.1.0 <2.9.0	Rigidi

2.2.5

La tabella seguente elenca le dipendenze per la versione 2.2.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.8.0	Rigidi
Greengrass CLI	>=2.1.0 <2.8.0	Rigidi

2.2.4

La tabella seguente elenca le dipendenze per la versione 2.2.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.7.0	Rigidi
Greengrass CLI	>=2.1.0 <2.7.0	Rigidi

2.2.3

La tabella seguente elenca le dipendenze per la versione 2.2.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.6.0	Rigidi
Greengrass CLI	>=2.1.0 <2.6.0	Rigidi

2.2.2

La tabella seguente elenca le dipendenze per la versione 2.2.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.5.0	Rigidi
Greengrass CLI	>=2.1.0 <2.5.0	Rigidi

2.2.1

La tabella seguente elenca le dipendenze per la versione 2.2.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.4.0	Rigidi

Dipendenza	Versioni compatibili	Tipo di dipendenza
Greengrass CLI	>=2.1.0 <2.4.0	Rigidi

2.2.0

La tabella seguente elenca le dipendenze per la versione 2.2.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.3.0	Rigidi
Greengrass CLI	>=2.1.0 <2.3.0	Rigidi

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.2.0	Rigidi
Greengrass CLI	>=2.1.0 <2.2.0	Rigidi

2.0.x

La tabella seguente elenca le dipendenze per la versione 2.0.x di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Flessibili
Greengrass CLI	>=2.0.3 <2.1.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

v2.1.x - v2.4.x

`httpsEnabled`

(Facoltativo) È possibile abilitare la comunicazione HTTPS per la console di debug locale. Se si abilita la comunicazione HTTPS, la console di debug locale crea un certificato autofirmato. I browser Web mostrano avvisi di sicurezza per i siti Web che utilizzano certificati autofirmati, pertanto è necessario verificare manualmente il certificato. È quindi possibile ignorare l'avviso. Per ulteriori informazioni, consulta Utilizzo.

Default: `true`

`port`

(Facoltativo) La porta alla quale fornire la console di debug locale.

Impostazione predefinita: `1441`

`websocketPort`

(Facoltativo) La porta websocket da utilizzare per la console di debug locale.

Impostazione predefinita: `1442`

`bindHostname`

(Facoltativo) Il nome host da utilizzare per la console di debug locale.

Se esegui il software AWS IoT Greengrass Core in un contenitore Docker, imposta questo parametro su, in modo da `0.0.0.0` poter aprire la console di debug locale all'esterno del contenitore Docker.

Impostazione predefinita: `localhost`

Example Esempio: aggiornamento basato sull'unione della configurazione

La configurazione di esempio seguente specifica di aprire la console di debug locale su porte non predefinite e disabilitare HTTPS.

```
{
  "httpsEnabled": false,
  "port": "10441",
  "websocketPort": "10442"
}
```

v2.0.x

`port`

(Facoltativo) La porta alla quale fornire la console di debug locale.

Impostazione predefinita: `1441`

`websocketPort`

(Facoltativo) La porta websocket da utilizzare per la console di debug locale.

Impostazione predefinita: `1442`

`bindHostname`

(Facoltativo) Il nome host da utilizzare per la console di debug locale.

Se esegui il software AWS IoT Greengrass Core in un contenitore Docker, imposta questo parametro su, in modo da `0.0.0.0` poter aprire la console di debug locale all'esterno del contenitore Docker.

Impostazione predefinita: `localhost`

Example Esempio: aggiornamento basato sull'unione della configurazione

La configurazione di esempio seguente specifica di aprire la console di debug locale su porte non predefinite.

```
{
  "port": "10441",
  "websocketPort": "10442"
}
```

Utilizzo

Per utilizzare la console di debug locale, create una sessione dalla CLI di Greengrass. Quando si crea una sessione, la CLI di Greengrass fornisce un nome utente e una password temporanea che è possibile utilizzare per accedere alla console di debug locale.

Segui queste istruzioni per aprire la console di debug locale sul tuo dispositivo principale o sul tuo computer di sviluppo.

v2.1.x - v2.4.x

Nelle versioni 2.1.0 e successive, la console di debug locale utilizza HTTPS per impostazione predefinita. Quando HTTPS è abilitato, la console di debug locale crea un certificato autofirmato per proteggere la connessione. Il browser Web mostra un avviso di sicurezza quando si apre la console di debug locale a causa di questo certificato autofirmato. Quando si crea una sessione con la Greengrass CLI, l'output include le impronte digitali del certificato, in modo da poter verificare che il certificato sia legittimo e che la connessione sia sicura.

È possibile disabilitare HTTPS. Per ulteriori informazioni, consulta Configurazione della console di debug locale.

Per aprire la console di debug locale

1. (Facoltativo) Per visualizzare la console di debug locale sul tuo computer di sviluppo, puoi inoltrare la porta della console tramite SSH. Tuttavia, è necessario prima abilitare l'`AllowTcpForwarding`opzione nel file di configurazione SSH del dispositivo principale. Per impostazione predefinita, questa opzione è abilitata. Esegui il comando seguente sul tuo computer di sviluppo per visualizzare la dashboard `localhost:1441` sul tuo computer di sviluppo.

   ```
   ssh -L 1441:localhost:1441 -L 1442:localhost:1442 username@core-device-ip-address
   ```

   > **Note**
   >
   > È possibile modificare le porte predefinite da `1441` e`1442`. Per ulteriori informazioni, vedere Configurazione della console di debug locale.

2. Crea una sessione per utilizzare la console di debug locale. Quando si crea una sessione, si genera una password da utilizzare per l'autenticazione. La console di debug locale richiede una password per aumentare la sicurezza, poiché è possibile utilizzare questo componente per visualizzare informazioni importanti ed eseguire operazioni sul dispositivo principale. La console di debug locale crea anche un certificato per proteggere la connessione se si abilita HTTPS nella configurazione del componente. HTTPS è abilitato per impostazione predefinita.

   Usa la AWS IoT Greengrass CLI per creare la sessione. Questo comando genera una password casuale di 43 caratteri che scade dopo 8 ore. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella principale. AWS IoT Greengrass V2

   Linux or Unix

   ```
   sudo /greengrass/v2/bin/greengrass-cli get-debug-password
   ```

   Windows

   ```
   C:\greengrass\v2\bin\greengrass-cli get-debug-password
   ```

   L'output del comando è simile al seguente esempio se la console di debug locale è stata configurata per l'utilizzo di HTTPS. Le impronte digitali del certificato vengono utilizzate per verificare che la connessione sia sicura quando si apre la console di debug locale.

   ```
   Username: debug
   Password: bEDp3MOHdj8ou2w5de_sCBI2XAaguy3a8XxREXAMPLE
   Password expires at: 2021-04-01T17:01:43.921999931-07:00
   The local debug console is configured to use TLS security. The certificate is
    self-signed so you will need to bypass your web browser's security warnings to
    open the console.
   Before you bypass the security warning, verify that the certificate fingerprint
    matches the following fingerprints.
   SHA-256: 15 0B 2C E2 54 8B 22 DE 08 46 54 8A B1 2B 25 DE FB 02 7D 01 4E 4A 56 67
    96 DA A6 CC B1 D2 C4 1B
   SHA-1: BC 3E 16 04 D3 80 70 DA E0 47 25 F9 90 FA D6 02 80 3E B5 C1
   ```

   Il componente debug view crea una sessione che dura 8 ore. Dopodiché, è necessario generare una nuova password per visualizzare nuovamente la console di debug locale.

3. Apri e accedi alla dashboard. Visualizza la dashboard sul tuo dispositivo principale Greengrass o sul tuo computer di sviluppo se inoltri la porta tramite SSH. Esegui una di queste operazioni:

   - Se hai abilitato HTTPS nella console di debug locale, che è l'impostazione predefinita, procedi come segue:

     a. Apri `https://localhost:1441` sul tuo dispositivo principale o sul tuo computer di sviluppo se hai inoltrato la porta tramite SSH.

        Il tuo browser potrebbe mostrare un avviso di sicurezza relativo a un certificato di sicurezza non valido.

     b. Se il tuo browser mostra un avviso di sicurezza, verifica che il certificato sia legittimo e ignora l'avviso di sicurezza. Esegui questa operazione:

        i. Trova l'impronta digitale SHA-256 o SHA-1 del certificato e verifica che corrisponda all'impronta digitale SHA-256 o SHA-1 stampata in precedenza dal comando. `get-debug-password` Il tuo browser potrebbe fornire una o entrambe le impronte digitali. Consulta la documentazione del tuo browser per visualizzare il certificato e le relative impronte digitali. In alcuni browser, l'impronta digitale del certificato viene chiamata impronta digitale.

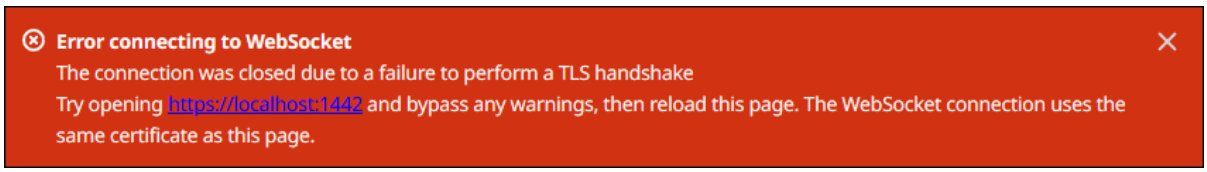
        > **Note**
        >
        > Se l'impronta digitale del certificato non corrisponde, vai Step 2 a creare una nuova sessione. Se l'impronta digitale del certificato continua a non corrispondere, la connessione potrebbe non essere sicura.

        ii. Se l'impronta digitale del certificato corrisponde, ignora l'avviso di sicurezza del browser per aprire la console di debug locale. Consultate la documentazione del browser per aggirare l'avviso di sicurezza del browser.

     c. Accedi al sito Web utilizzando il nome utente e la password stampati in precedenza con il `get-debug-password` comando.

        Si apre la console di debug locale.

     d. Se la console di debug locale mostra un errore che indica che non è possibile connettersi a the a WebSocket causa di un handshake TLS non riuscito, è necessario ignorare l'avviso di sicurezza autofirmato per l'URL. WebSocket



Esegui questa operazione:

i. Apri `https://localhost:1442` nello stesso browser in cui hai aperto la console di debug locale.

ii. Verifica il certificato e ignora l'avviso di sicurezza.

Il browser potrebbe mostrare una pagina HTTP 404 dopo aver ignorato l'avviso.

iii. `https://localhost:1441`Riapri.

La console di debug locale mostra informazioni sul dispositivo principale.

- Se hai disabilitato HTTPS nella console di debug locale, procedi come segue:

a. Apri `http://localhost:1441` sul tuo dispositivo principale o sul tuo computer di sviluppo se hai inoltrato la porta tramite SSH.

b. Accedi al sito Web utilizzando il nome utente e la password stampati in precedenza con il `get-debug-password` comando.

Si apre la console di debug locale.

v2.0.x

Per aprire la console di debug locale

1. (Facoltativo) Per visualizzare la console di debug locale sul tuo computer di sviluppo, puoi inoltrare la porta della console tramite SSH. Tuttavia, è necessario prima abilitare l'`AllowTcpForwarding`opzione nel file di configurazione SSH del dispositivo principale. Per impostazione predefinita, questa opzione è abilitata. Esegui il comando seguente sul tuo computer di sviluppo per visualizzare la dashboard `localhost:1441` sul tuo computer di sviluppo.

```
ssh -L 1441:localhost:1441 -L 1442:localhost:1442 username@core-device-ip-address
```

> **Note**
>
> È possibile modificare le porte predefinite da `1441` e`1442`. Per ulteriori informazioni, vedere Configurazione della console di debug locale.

2. Crea una sessione per utilizzare la console di debug locale. Quando si crea una sessione, si genera una password da utilizzare per l'autenticazione. La console di debug locale richiede una password per aumentare la sicurezza, poiché è possibile utilizzare questo componente per visualizzare informazioni importanti ed eseguire operazioni sul dispositivo principale.

   Usa la AWS IoT Greengrass CLI per creare la sessione. Questo comando genera una password casuale di 43 caratteri che scade dopo 8 ore. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella principale. AWS IoT Greengrass V2

   Linux or Unix

   ```
   sudo /greengrass/v2/bin/greengrass-cli get-debug-password
   ```

   Windows

   ```
   C:\greengrass\v2\bin\greengrass-cli get-debug-password
   ```

   L'output del comando è simile all'esempio seguente.

   ```
   Username: debug
   Password: bEDp3MOHdj8ou2w5de_sCBI2XAaguy3a8XxREXAMPLE
   Password will expire at: 2021-04-01T17:01:43.921999931-07:00
   ```

   Il componente debug view crea una sessione della durata di 4 ore, quindi è necessario generare una nuova password per visualizzare nuovamente la console di debug locale.

3. Apri `http://localhost:1441` sul tuo dispositivo principale o sul tuo computer di sviluppo se hai inoltrato la porta tramite SSH.

4. Accedi al sito Web utilizzando il nome utente e la password stampati in precedenza con il `get-debug-password` comando.

   Si apre la console di debug locale.

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
2.4.2	Correzioni di bug e miglioramenti • Correzioni di bug generali e miglioramenti.
2.4.1	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.

Versione	Modifiche
2.4.0	Nuove funzionalità • Aggiunge la console di debug dello stream manager.
2.3.1	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.3.0	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus. Nuove funzionalità • Include un client di debug PubSub MQTT AWS IoT Core .
2.2.7	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.2.6	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.2.5	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.2.4	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.2.3	Correzioni di bug e miglioramenti • Risolve un problema che impediva l'avvio quando il componente non riusciva a decrittografare il keystore che contiene la chiave privata SSL. • Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.2.2	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.2.1	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.2.0	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.1.0	Nuove funzionalità • Utilizza HTTPS per proteggere la connessione alla console di debug locale. HTTPS è abilitato per impostazione predefinita. Correzioni di bug e miglioramenti • Puoi eliminare i messaggi della flashbar nell'editor di configurazione.
2.0.3	Versione iniziale.

Gestore dei registri

Il componente log manager (`aws.greengrass.LogManager`) carica i log dai dispositivi AWS IoT Greengrass principali su Amazon CloudWatch Logs. Puoi caricare i log dal nucleo di Greengrass, da altri componenti Greengrass e da altre applicazioni e servizi che non sono componenti Greengrass. Per ulteriori informazioni su come monitorare i log in CloudWatch Logs e nel file system locale, vedere. Monitora AWS IoT Greengrass i registri

Le seguenti considerazioni si applicano quando si utilizza il componente log manager per scrivere nei registri: CloudWatch

- Registra i ritardi

> **Note**
>
> Si consiglia di eseguire l'aggiornamento alla versione 2.3.0 di log manager, che riduce i ritardi di registro per i file di registro ruotati e attivi. Quando esegui l'aggiornamento a log manager 2.3.0, ti consigliamo di eseguire anche l'aggiornamento a Greengrass nucleus 2.9.1.

Il componente di log manager versione 2.2.8 (e precedenti) elabora e carica i log solo dai file di registro ruotati. Per impostazione predefinita, il software AWS IoT Greengrass Core ruota i file di registro ogni ora o dopo che hanno raggiunto i 1.024 KB. Di conseguenza, il componente log manager carica i log solo dopo che il software AWS IoT Greengrass Core o un componente Greengrass hanno scritto oltre 1.024 KB di log. È possibile configurare un limite inferiore per le dimensioni dei file di registro per far sì che i file di registro ruotino più spesso. Ciò fa sì che il componente log manager carichi i log in Logs più CloudWatch frequentemente.

Il componente di gestione dei registri versione 2.3.0 (e successive) elabora e carica tutti i log. Quando si scrive un nuovo registro, la versione 2.3.0 (e successive) di log manager elabora e carica direttamente il file di registro attivo anziché attendere che venga ruotato. Ciò significa che è possibile visualizzare il nuovo registro in 5 minuti o meno.

Il componente di gestione dei registri carica periodicamente nuovi registri. Per impostazione predefinita, il componente di gestione dei registri carica nuovi registri ogni 5 minuti. È possibile configurare un intervallo di caricamento inferiore, in modo che il componente log manager carichi i

log in Logs più frequentemente configurando CloudWatch . `periodicUploadIntervalSec` Per ulteriori informazioni su come configurare questo intervallo periodico, vedere Configurazione.

I log possono essere caricati quasi in tempo reale dallo stesso file system Greengrass. Se hai bisogno di osservare i log in tempo reale, prendi in considerazione l'utilizzo dei log del file system.

> **Note**
>
> Se utilizzi file system diversi su cui scrivere i log, log manager torna al comportamento delle versioni 2.2.8 e precedenti dei componenti di log manager. Per informazioni sull'accesso ai log del file system, consultate Access file system logs.

- Inclinazione dell'orologio

  Il componente log manager utilizza il processo di firma standard Signature Version 4 per creare richieste API per CloudWatch Logs. Se l'ora del sistema su un dispositivo principale non è sincronizzata per più di 15 minuti, CloudWatch Logs rifiuta le richieste. Per ulteriori informazioni, consulta la pagina relativa al processo di firma Signature Version 4 nella Riferimenti generali di AWS.

Per informazioni sui gruppi di log e sui flussi di log in cui questo componente carica i log, consulta. Utilizzo

Argomenti

- Versioni
- Type
- Sistema operativo
- Requisiti
- Dipendenze
- Configurazione
- Utilizzo
- File di registro locale
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 2.3.x
- 2.2.x
- 2.1.x
- 2,0x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Il ruolo del dispositivo Greengrass deve consentire le `logs:DescribeLogStreams` azioni`logs:CreateLogGroup`,`logs:CreateLogStream`, `elogs:PutLogEvents`,, come illustrato nel seguente esempio di policy IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

> **Note**
>
> Il ruolo del dispositivo Greengrass che crei quando installi il software AWS IoT Greengrass Core include per impostazione predefinita le autorizzazioni di questo criterio di esempio.

Per ulteriori informazioni, consulta Using Identity-Based Policy (IAM policies) for CloudWatch Logs nella Amazon CloudWatch Logs User Guide.

- Il componente log manager è supportato per l'esecuzione in un VPC. Per distribuire questo componente in un VPC, è necessario quanto segue.
  - Il componente di gestione dei registri deve disporre di una connettività a `logs.region.amazonaws.com` cui abbia l'endpoint VPC di. `com.amazonaws.us-east-1.logs`

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`logs.region.amazonaws.com`	443	No	Obbligatorio se si

Endpoint	Porta	Richiesto	Descrizione
			scrivono log in Logs. CloudWatch

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.3.7

La tabella seguente elenca le dipendenze per la versione 2.3.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.13.0	Flessibili

2.3.5 and 2.3.6

La tabella seguente elenca le dipendenze per le versioni 2.3.5 e 2.3.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.12.0	Flessibili

2.3.3 – 2.3.4

La tabella seguente elenca le dipendenze per le versioni da 2.3.3 a 2.3.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.11.0	Flessibili

2.2.8 – 2.3.2

La tabella seguente elenca le dipendenze per le versioni da 2.2.8 a 2.3.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.10.0	Flessibili

2.2.7

La tabella seguente elenca le dipendenze per la versione 2.2.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.9.0	Flessibili

2.2.6

La tabella seguente elenca le dipendenze per la versione 2.2.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.8.0	Flessibili

2.2.5

La tabella seguente elenca le dipendenze per la versione 2.2.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.7.0	Flessibili

2.2.1 - 2.2.4

La tabella seguente elenca le dipendenze per le versioni 2.2.1 - 2.2.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.6.0	Flessibili

2.1.3 and 2.2.0

La tabella seguente elenca le dipendenze per le versioni 2.1.3 e 2.2.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.5.0	Flessibili

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.4.0	Flessibili

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.3.0	Flessibili

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.1.0 <2.2.0	Flessibili

2.0.x

La tabella seguente elenca le dipendenze per la versione 2.0.x di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

v2.3.6 – v2.3.7

`logsUploaderConfiguration`

(Facoltativo) La configurazione per i log caricati dal componente Log Manager. Questo oggetto contiene le seguenti informazioni:

`systemLogsConfiguration`

(Facoltativo) La configurazione per i registri di sistema del software AWS IoT Greengrass Core, che includono i registri del nucleo Greengrass e i componenti del plug-in. Specificate questa configurazione per consentire al componente di gestione dei log di gestire i log di sistema. Questo oggetto contiene le seguenti informazioni:

`uploadToCloudWatch`

(Facoltativo) È possibile caricare i log di sistema in CloudWatch Logs.

Impostazione predefinita: `false`

`minimumLogLevel`

(Facoltativo) Il livello minimo di messaggi di registro da caricare. Questo livello minimo si applica solo se si configura il componente Greengrass nucleus per l'output di log in formato JSON. Per abilitare i log in formato JSON, specificate il parametro del formato JSON di registrazione (). `logging.format`

Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`diskSpaceLimit`

(Facoltativo) La dimensione totale massima dei file di registro del sistema Greengrass, nell'unità specificata. `diskSpaceLimitUnit` Dopo che la dimensione totale dei file di registro di sistema Greengrass supera questa dimensione totale massima, il software AWS IoT Greengrass Core elimina i file di registro di sistema Greengrass più vecchi.

Questo parametro è equivalente al parametro log size limit (`totalLogsSizeKB`) del componente Greengrass nucleus. Il software AWS IoT Greengrass Core utilizza il minimo dei due valori come dimensione massima totale del registro di sistema Greengrass.

`diskSpaceLimitUnit`

(Opzionale) L'unità per. `diskSpaceLimit` Seleziona una delle opzioni seguenti:

- `KB`— kilobyte
- `MB`— megabyte
- `GB`— gigabyte

Impostazione predefinita: `KB`

`deleteLogFileAfterCloudUpload`

(Facoltativo) È possibile eliminare un file di registro dopo che il componente di gestione dei registri ha caricato i registri in Logs. CloudWatch

Impostazione predefinita: `false`

`componentLogsConfigurationMap`

(Facoltativo) Una mappa delle configurazioni dei log per i componenti del dispositivo principale. Ogni `componentName` oggetto in questa mappa definisce la configurazione del registro per il componente o l'applicazione. Il componente log manager carica questi registri dei componenti in Logs. CloudWatch

> **Important**
>
> Consigliamo vivamente di utilizzare una sola chiave di configurazione per componente. È consigliabile scegliere come target solo un gruppo di file con un solo file di registro in cui viene scritto attivamente quando si utilizza`logFileRegex`. La mancata osservanza di questa raccomandazione può comportare il caricamento di log duplicati in. CloudWatch Se hai come target più file di log attivi con un'unica espressione regolare, ti consigliamo di eseguire l'aggiornamento a log manager v2.3.1 o versione successiva e di prendere in considerazione la possibilità di modificare la configurazione utilizzando la configurazione di esempio.

> **Note**
>
> Se stai eseguendo l'aggiornamento da una versione di log manager precedente alla v2.2.0, puoi continuare a utilizzare l'elenco anziché. `componentLogsConfiguration` `componentLogsConfigurationMap` Tuttavia, ti consigliamo vivamente di utilizzare il formato della mappa in modo da poter utilizzare gli aggiornamenti di unione e ripristino per modificare le configurazioni di componenti specifici. Per informazioni sul `componentLogsConfiguration` parametro, consulta i parametri di configurazione per la versione 2.1.x di questo componente.

`componentName`

La configurazione del registro per il *`componentName`*componente o l'applicazione per questa configurazione di registro. È possibile specificare il nome di un componente Greengrass o un altro valore per identificare questo gruppo di log.

Ogni oggetto contiene le seguenti informazioni:

`minimumLogLevel`

(Facoltativo) Il livello minimo di messaggi di registro da caricare. Questo livello minimo si applica solo se i log di questo componente utilizzano un formato JSON specifico, disponibile nel repository del modulo di AWS IoT Greengrass registrazione su. GitHub

Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`diskSpaceLimit`

(Facoltativo) La dimensione totale massima di tutti i file di registro per questo componente, nell'unità specificata`diskSpaceLimitUnit`. Dopo che la dimensione totale dei file di registro di questo componente supera questa dimensione totale massima, il software AWS IoT Greengrass Core elimina i file di registro più vecchi di questo componente.

Questo parametro è correlato al parametro limite di dimensione del registro (`totalLogsSizeKB`) del componente Greengrass nucleus. Il software AWS IoT Greengrass Core utilizza il minimo dei due valori come dimensione massima totale del registro per questo componente.

`diskSpaceLimitUnit`

(Facoltativo) L'unità per`diskSpaceLimit`. Seleziona una delle opzioni seguenti:

- `KB`— kilobyte
- `MB`— megabyte
- `GB`— gigabyte

Impostazione predefinita: `KB`

`logFileDirectoryPath`

(Facoltativo) Il percorso della cartella che contiene i file di registro di questo componente.

Non è necessario specificare questo parametro per i componenti Greengrass che stampano su standard output (stdout) e standard error (stderr).

Default: */greengrass/v2*`/logs`.

`logFileRegex`

(Facoltativo) Un'espressione regolare che specifica il formato del nome del file di registro utilizzato dal componente o dall'applicazione. Il componente log manager utilizza questa espressione regolare per identificare i file di registro nella cartella in`logFileDirectoryPath`.

Non è necessario specificare questo parametro per i componenti Greengrass che stampano su standard output (stdout) e standard error (stderr).

Se il componente o l'applicazione ruota i file di registro, specificate un'espressione regolare che corrisponda ai nomi dei file di registro ruotati. Ad esempio, potresti specificare di caricare i log **`hello_world\\\\w*.log`** per un'applicazione Hello World. Il `\\\\w*` modello corrisponde a zero o più caratteri di parola, inclusi caratteri alfanumerici e caratteri di sottolineatura. Questa espressione regolare corrisponde ai file di registro con e senza timestamp nel nome. In questo esempio, il gestore dei log carica i seguenti file di registro:

- `hello_world.log`— Il file di registro più recente per l'applicazione Hello World.
- `hello_world_2020_12_15_17_0.log`— Un vecchio file di registro per l'applicazione Hello World.

Impostazione predefinita:*componentName*`\\\\w*.log`, dove *componentName* è il nome del componente per questa configurazione di registro.

`deleteLogFileAfterCloudUpload`

(Facoltativo) È possibile eliminare un file di registro dopo che il componente di gestione dei registri ha caricato i log in Logs. CloudWatch

Impostazione predefinita: `false`

`multiLineStartPattern`

(Facoltativo) Un'espressione regolare che identifica quando un messaggio di registro su una nuova riga è un nuovo messaggio di registro. Se l'espressione regolare non corrisponde alla nuova riga, il componente log manager aggiunge la nuova riga al messaggio di registro della riga precedente.

Per impostazione predefinita, il componente log manager controlla se la riga inizia con uno spazio bianco, ad esempio una tabulazione o uno spazio. In caso contrario, il gestore dei registri gestisce quella riga come un nuovo messaggio di registro. Altrimenti, aggiunge quella riga al messaggio di registro corrente. Questo comportamento garantisce che il componente di gestione dei log non divida i messaggi che si estendono su più righe, come le tracce dello stack.

`periodicUploadIntervalSec`

(Facoltativo) Il periodo in secondi in cui il componente di gestione dei registri verifica la presenza di nuovi file di registro da caricare.

Impostazione predefinita: `300` (5 minuti)

Minimo: `0.000001` (1 microsecondo)

`deprecatedVersionSupport`

Indica se il gestore dei registri deve utilizzare i miglioramenti della velocità di registrazione introdotti in log manager v2.3.5. Imposta il valore su cui utilizzare `false` i miglioramenti.

Se si imposta questo valore su `false` quando si esegue l'aggiornamento da log manager v2.3.1 o versioni precedenti, è possibile che vengano caricate voci di registro duplicate.

Il valore predefinito è `true`.

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica di caricare i registri di sistema e i registri dei `com.example.HelloWorld` componenti in Logs. CloudWatch

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
```

```
      "uploadToCloudWatch": "true",
      "minimumLogLevel": "INFO",
      "diskSpaceLimit": "10",
      "diskSpaceLimitUnit": "MB",
      "deleteLogFileAfterCloudUpload": "false"
    },
    "componentLogsConfigurationMap": {
      "com.example.HelloWorld": {
        "minimumLogLevel": "INFO",
        "diskSpaceLimit": "20",
        "diskSpaceLimitUnit": "MB",
        "deleteLogFileAfterCloudUpload": "false"
      }
    }
  },
  "periodicUploadIntervalSec": "300",
  "deprecatedVersionSupport": "false"
}
```

Example Esempio: configurazione per caricare più file di registro attivi utilizzando log manager v2.3.1

La configurazione di esempio seguente è l'esempio consigliato se si desidera indirizzare più file di registro attivi. Questo esempio di configurazione specifica in quali file di registro attivi si desidera caricare CloudWatch. Utilizzando questo esempio di configurazione, la configurazione caricherà anche tutti i file ruotati che corrispondono a. `logFileRegex` Questa configurazione di esempio è supportata in log manager v2.3.1.

```
{
  "logsUploaderConfiguration": {
    "componentLogsConfigurationMap": {
      "com.example.A": {
        "logFileRegex": "com.example.A\\w*.log",
        "deleteLogFileAfterCloudUpload": "false"
      }
      "com.example.B": {
        "logFileRegex": "com.example.B\\w*.log",
        "deleteLogFileAfterCloudUpload": "false"
      }
    }
  },
  "periodicUploadIntervalSec": "10"
```

```
}
```

v2.3.x

`logsUploaderConfiguration`

(Facoltativo) La configurazione per i log caricati dal componente log manager. Questo oggetto contiene le seguenti informazioni:

`systemLogsConfiguration`

(Facoltativo) La configurazione per i registri di sistema del software AWS IoT Greengrass Core, che includono i registri del nucleo Greengrass e i componenti del plug-in. Specificate questa configurazione per consentire al componente di gestione dei log di gestire i log di sistema. Questo oggetto contiene le seguenti informazioni:

`uploadToCloudWatch`

(Facoltativo) È possibile caricare i log di sistema in CloudWatch Logs.

Impostazione predefinita: `false`

`minimumLogLevel`

(Facoltativo) Il livello minimo di messaggi di registro da caricare. Questo livello minimo si applica solo se si configura il componente Greengrass nucleus per l'output di log in formato JSON. Per abilitare i log in formato JSON, specificate il parametro del formato JSON di registrazione (). `logging.format`

Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`diskSpaceLimit`

(Facoltativo) La dimensione totale massima dei file di registro del sistema Greengrass, nell'unità specificata. `diskSpaceLimitUnit` Dopo che la dimensione totale dei file di

registro di sistema Greengrass supera questa dimensione totale massima, il software AWS IoT Greengrass Core elimina i file di registro di sistema Greengrass più vecchi.

Questo parametro è equivalente al parametro log size limit (`totalLogsSizeKB`) del componente Greengrass nucleus. Il software AWS IoT Greengrass Core utilizza il minimo dei due valori come dimensione massima totale del registro di sistema Greengrass.

`diskSpaceLimitUnit`

(Opzionale) L'unità per. `diskSpaceLimit` Seleziona una delle opzioni seguenti:

- `KB`— kilobyte
- `MB`— megabyte
- `GB`— gigabyte

Impostazione predefinita: `KB`

`deleteLogFileAfterCloudUpload`

(Facoltativo) È possibile eliminare un file di registro dopo che il componente di gestione dei registri ha caricato i registri in Logs. CloudWatch

Impostazione predefinita: `false`

`componentLogsConfigurationMap`

(Facoltativo) Una mappa delle configurazioni dei log per i componenti del dispositivo principale. Ogni `componentName` oggetto in questa mappa definisce la configurazione del registro per il componente o l'applicazione. Il componente log manager carica questi registri dei componenti in Logs. CloudWatch

> **Important**
>
> Consigliamo vivamente di utilizzare una sola chiave di configurazione per componente. È consigliabile scegliere come target solo un gruppo di file con un solo file di registro in cui viene scritto attivamente quando si utilizza`logFileRegex`. La mancata osservanza di questa raccomandazione può comportare il caricamento di log duplicati in. CloudWatch Se hai come target più file di log attivi con un'unica espressione regolare, ti consigliamo di eseguire l'aggiornamento a log manager v2.3.1 e di prendere in considerazione la possibilità di modificare la configurazione utilizzando la configurazione di esempio.

> **Note**
>
> Se stai eseguendo l'aggiornamento da una versione di log manager precedente alla v2.2.0, puoi continuare a utilizzare l'elenco anziché. `componentLogsConfiguration` `componentLogsConfigurationMap` Tuttavia, ti consigliamo vivamente di utilizzare il formato della mappa in modo da poter utilizzare gli aggiornamenti di unione e ripristino per modificare le configurazioni di componenti specifici. Per informazioni sul `componentLogsConfiguration` parametro, consulta i parametri di configurazione per la versione 2.1.x di questo componente.

`componentName`

La configurazione del registro per il *`componentName`*componente o l'applicazione per questa configurazione di registro. È possibile specificare il nome di un componente Greengrass o un altro valore per identificare questo gruppo di log.

Ogni oggetto contiene le seguenti informazioni:

`minimumLogLevel`

(Facoltativo) Il livello minimo di messaggi di registro da caricare. Questo livello minimo si applica solo se i log di questo componente utilizzano un formato JSON specifico, disponibile nel repository del modulo di AWS IoT Greengrass registrazione su. GitHub

Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`diskSpaceLimit`

(Facoltativo) La dimensione totale massima di tutti i file di registro per questo componente, nell'unità specificata`diskSpaceLimitUnit`. Dopo che la dimensione

totale dei file di registro di questo componente supera questa dimensione totale massima, il software AWS IoT Greengrass Core elimina i file di registro più vecchi di questo componente.

Questo parametro è correlato al parametro limite di dimensione del registro (`totalLogsSizeKB`) del componente Greengrass nucleus. Il software AWS IoT Greengrass Core utilizza il minimo dei due valori come dimensione massima totale del registro per questo componente.

`diskSpaceLimitUnit`

(Facoltativo) L'unità per`diskSpaceLimit`. Seleziona una delle opzioni seguenti:

- `KB`— kilobyte
- `MB`— megabyte
- `GB`— gigabyte

Impostazione predefinita: `KB`

`logFileDirectoryPath`

(Facoltativo) Il percorso della cartella che contiene i file di registro di questo componente.

Non è necessario specificare questo parametro per i componenti Greengrass che stampano su standard output (stdout) e standard error (stderr).

Default: */greengrass/v2*`/logs`.

`logFileRegex`

(Facoltativo) Un'espressione regolare che specifica il formato del nome del file di registro utilizzato dal componente o dall'applicazione. Il componente log manager utilizza questa espressione regolare per identificare i file di registro nella cartella in`logFileDirectoryPath`.

Non è necessario specificare questo parametro per i componenti Greengrass che stampano su standard output (stdout) e standard error (stderr).

Se il componente o l'applicazione ruota i file di registro, specificate un'espressione regolare che corrisponda ai nomi dei file di registro ruotati. Ad esempio, potresti

specificare di caricare i log **`hello_world\\\\w*.log`** per un'applicazione Hello World. Il `\\\\w*` modello corrisponde a zero o più caratteri di parola, inclusi caratteri alfanumerici e caratteri di sottolineatura. Questa espressione regolare corrisponde ai file di registro con e senza timestamp nel nome. In questo esempio, il gestore dei log carica i seguenti file di registro:

- `hello_world.log`— Il file di registro più recente per l'applicazione Hello World.
- `hello_world_2020_12_15_17_0.log`— Un vecchio file di registro per l'applicazione Hello World.

Impostazione predefinita:*`componentName`*`\\\\w*.log`, dove *`componentName`* è il nome del componente per questa configurazione di registro.

`deleteLogFileAfterCloudUpload`

(Facoltativo) È possibile eliminare un file di registro dopo che il componente di gestione dei registri ha caricato i log in Logs. CloudWatch

Impostazione predefinita: `false`

`multiLineStartPattern`

(Facoltativo) Un'espressione regolare che identifica quando un messaggio di registro su una nuova riga è un nuovo messaggio di registro. Se l'espressione regolare non corrisponde alla nuova riga, il componente log manager aggiunge la nuova riga al messaggio di registro della riga precedente.

Per impostazione predefinita, il componente log manager controlla se la riga inizia con uno spazio bianco, ad esempio una tabulazione o uno spazio. In caso contrario, il gestore dei registri gestisce quella riga come un nuovo messaggio di registro. Altrimenti, aggiunge quella riga al messaggio di registro corrente. Questo comportamento garantisce che il componente di gestione dei log non divida i messaggi che si estendono su più righe, come le tracce dello stack.

`periodicUploadIntervalSec`

(Facoltativo) Il periodo in secondi in cui il componente di gestione dei registri verifica la presenza di nuovi file di registro da caricare.

Impostazione predefinita: `300` (5 minuti)

Minimo: `0.000001` (1 microsecondo)

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica di caricare i registri di sistema e i registri dei `com.example.HelloWorld` componenti in Logs. CloudWatch

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true",
      "minimumLogLevel": "INFO",
      "diskSpaceLimit": "10",
      "diskSpaceLimitUnit": "MB",
      "deleteLogFileAfterCloudUpload": "false"
    },
    "componentLogsConfigurationMap": {
      "com.example.HelloWorld": {
        "minimumLogLevel": "INFO",
        "diskSpaceLimit": "20",
        "diskSpaceLimitUnit": "MB",
        "deleteLogFileAfterCloudUpload": "false"
      }
    }
  },
  "periodicUploadIntervalSec": "300"
}
```

Example Esempio: configurazione per caricare più file di registro attivi utilizzando log manager v2.3.1

La configurazione di esempio seguente è l'esempio consigliato se si desidera indirizzare più file di registro attivi. Questo esempio di configurazione specifica in quali file di registro attivi si desidera caricare CloudWatch. Utilizzando questo esempio di configurazione, la configurazione caricherà anche tutti i file ruotati che corrispondono a. `logFileRegex` Questa configurazione di esempio è supportata in log manager v2.3.1.

```
{
  "logsUploaderConfiguration": {
    "componentLogsConfigurationMap": {
      "com.example.A": {
        "logFileRegex": "com.example.A\\w*.log",
        "deleteLogFileAfterCloudUpload": "false"
      }
```

```
      "com.example.B": {
        "logFileRegex": "com.example.B\\w*.log",
        "deleteLogFileAfterCloudUpload": "false"
      }
    }
  },
  "periodicUploadIntervalSec": "10"
}
```

v2.2.x

`logsUploaderConfiguration`

(Facoltativo) La configurazione per i log caricati dal componente log manager. Questo oggetto contiene le seguenti informazioni:

`systemLogsConfiguration`

(Facoltativo) La configurazione per i registri di sistema del software AWS IoT Greengrass Core, che includono i registri del nucleo Greengrass e i componenti del plug-in. Specificate questa configurazione per consentire al componente di gestione dei log di gestire i log di sistema. Questo oggetto contiene le seguenti informazioni:

`uploadToCloudWatch`

(Facoltativo) È possibile caricare i log di sistema in CloudWatch Logs.

Impostazione predefinita: `false`

`minimumLogLevel`

(Facoltativo) Il livello minimo di messaggi di registro da caricare. Questo livello minimo si applica solo se si configura il componente Greengrass nucleus per l'output di log in formato JSON. Per abilitare i log in formato JSON, specificate il parametro del formato JSON di registrazione (). `logging.format`

Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`diskSpaceLimit`

(Facoltativo) La dimensione totale massima dei file di registro del sistema Greengrass, nell'unità specificata. `diskSpaceLimitUnit` Dopo che la dimensione totale dei file di registro di sistema Greengrass supera questa dimensione totale massima, il software AWS IoT Greengrass Core elimina i file di registro di sistema Greengrass più vecchi.

Questo parametro è equivalente al parametro log size limit (`totalLogsSizeKB`) del componente Greengrass nucleus. Il software AWS IoT Greengrass Core utilizza il minimo dei due valori come dimensione massima totale del registro di sistema Greengrass.

`diskSpaceLimitUnit`

(Opzionale) L'unità per. `diskSpaceLimit` Seleziona una delle opzioni seguenti:

- `KB`— kilobyte
- `MB`— megabyte
- `GB`— gigabyte

Impostazione predefinita: `KB`

`deleteLogFileAfterCloudUpload`

(Facoltativo) È possibile eliminare un file di registro dopo che il componente di gestione dei registri ha caricato i registri in Logs. CloudWatch

Impostazione predefinita: `false`

`componentLogsConfigurationMap`

(Facoltativo) Una mappa delle configurazioni dei log per i componenti del dispositivo principale. Ogni `componentName` oggetto in questa mappa definisce la configurazione del registro per il componente o l'applicazione. Il componente log manager carica questi registri dei componenti in Logs. CloudWatch

> **Note**
>
> Se stai eseguendo l'aggiornamento da una versione di log manager precedente alla v2.2.0, puoi continuare a utilizzare l'elenco anziché. `componentLogsConfiguration` `componentLogsConfigurationMap` Tuttavia, ti consigliamo vivamente di utilizzare il formato della mappa in modo da poter utilizzare gli aggiornamenti di unione e ripristino per

modificare le configurazioni di componenti specifici. Per informazioni sul `componentLogsConfiguration` parametro, consulta i parametri di configurazione per la versione 2.1.x di questo componente.

`componentName`

La configurazione del registro per il *`componentName`*componente o l'applicazione per questa configurazione di registro. È possibile specificare il nome di un componente Greengrass o un altro valore per identificare questo gruppo di log.

Ogni oggetto contiene le seguenti informazioni:

`minimumLogLevel`

(Facoltativo) Il livello minimo di messaggi di registro da caricare. Questo livello minimo si applica solo se i log di questo componente utilizzano un formato JSON specifico, disponibile nel repository del modulo di AWS IoT Greengrass registrazione su. GitHub

Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`diskSpaceLimit`

(Facoltativo) La dimensione totale massima di tutti i file di registro per questo componente, nell'unità specificata`diskSpaceLimitUnit`. Dopo che la dimensione totale dei file di registro di questo componente supera questa dimensione totale massima, il software AWS IoT Greengrass Core elimina i file di registro più vecchi di questo componente.

Questo parametro è correlato al parametro limite di dimensione del registro (`totalLogsSizeKB`) del componente Greengrass nucleus. Il software AWS IoT Greengrass Core utilizza il minimo dei due valori come dimensione massima totale del registro per questo componente.

`diskSpaceLimitUnit`

(Facoltativo) L'unità per`diskSpaceLimit`. Seleziona una delle opzioni seguenti:

- `KB`— kilobyte
- `MB`— megabyte
- `GB`— gigabyte

Impostazione predefinita: `KB`

`logFileDirectoryPath`

(Facoltativo) Il percorso della cartella che contiene i file di registro di questo componente.

Non è necessario specificare questo parametro per i componenti Greengrass che stampano su standard output (stdout) e standard error (stderr).

Default: */greengrass/v2*`/logs`.

`logFileRegex`

(Facoltativo) Un'espressione regolare che specifica il formato del nome del file di registro utilizzato dal componente o dall'applicazione. Il componente log manager utilizza questa espressione regolare per identificare i file di registro nella cartella in`logFileDirectoryPath`.

Non è necessario specificare questo parametro per i componenti Greengrass che stampano su standard output (stdout) e standard error (stderr).

Se il componente o l'applicazione ruota i file di registro, specificate un'espressione regolare che corrisponda ai nomi dei file di registro ruotati. Ad esempio, potresti specificare di caricare i log **`hello_world\\\\w*.log`** per un'applicazione Hello World. Il `\\\\w*` modello corrisponde a zero o più caratteri di parola, inclusi caratteri alfanumerici e caratteri di sottolineatura. Questa espressione regolare corrisponde ai file di registro con e senza timestamp nel nome. In questo esempio, il gestore dei log carica i seguenti file di registro:

- `hello_world.log`— Il file di registro più recente per l'applicazione Hello World.
- `hello_world_2020_12_15_17_0.log`— Un vecchio file di registro per l'applicazione Hello World.

Impostazione predefinita:*componentName*\\\\w*.log, dove *componentName* è il nome del componente per questa configurazione di registro.

`deleteLogFileAfterCloudUpload`

(Facoltativo) È possibile eliminare un file di registro dopo che il componente di gestione dei registri ha caricato i log in Logs. CloudWatch

Impostazione predefinita: `false`

`multiLineStartPattern`

(Facoltativo) Un'espressione regolare che identifica quando un messaggio di registro su una nuova riga è un nuovo messaggio di registro. Se l'espressione regolare non corrisponde alla nuova riga, il componente log manager aggiunge la nuova riga al messaggio di registro della riga precedente.

Per impostazione predefinita, il componente log manager controlla se la riga inizia con uno spazio bianco, ad esempio una tabulazione o uno spazio. In caso contrario, il gestore dei registri gestisce quella riga come un nuovo messaggio di registro. Altrimenti, aggiunge quella riga al messaggio di registro corrente. Questo comportamento garantisce che il componente di gestione dei log non divida i messaggi che si estendono su più righe, come le tracce dello stack.

`periodicUploadIntervalSec`

(Facoltativo) Il periodo in secondi in cui il componente di gestione dei registri verifica la presenza di nuovi file di registro da caricare.

Impostazione predefinita: `300` (5 minuti)

Minimo: `0.000001` (1 microsecondo)

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica di caricare i registri di sistema e i registri dei `com.example.HelloWorld` componenti in Logs. CloudWatch

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true",
```

```
      "minimumLogLevel": "INFO",
      "diskSpaceLimit": "10",
      "diskSpaceLimitUnit": "MB",
      "deleteLogFileAfterCloudUpload": "false"
    },
    "componentLogsConfigurationMap": {
      "com.example.HelloWorld": {
        "minimumLogLevel": "INFO",
        "diskSpaceLimit": "20",
        "diskSpaceLimitUnit": "MB",
        "deleteLogFileAfterCloudUpload": "false"
      }
    }
  },
  "periodicUploadIntervalSec": "300"
}
```

v2.1.x

`logsUploaderConfiguration`

(Facoltativo) La configurazione per i log caricati dal componente Log Manager. Questo oggetto contiene le seguenti informazioni:

`systemLogsConfiguration`

(Facoltativo) La configurazione per i registri di sistema del software AWS IoT Greengrass Core, che includono i registri del nucleo Greengrass e i componenti del plug-in. Specificate questa configurazione per consentire al componente di gestione dei log di gestire i log di sistema. Questo oggetto contiene le seguenti informazioni:

`uploadToCloudWatch`

(Facoltativo) È possibile caricare i log di sistema in CloudWatch Logs.

Impostazione predefinita: `false`

`minimumLogLevel`

(Facoltativo) Il livello minimo di messaggi di registro da caricare. Questo livello minimo si applica solo se si configura il componente Greengrass nucleus per l'output di log in formato JSON. Per abilitare i log in formato JSON, specificate il parametro del formato JSON di registrazione (). `logging.format`

Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`diskSpaceLimit`

(Facoltativo) La dimensione totale massima dei file di registro del sistema Greengrass, nell'unità specificata. `diskSpaceLimitUnit` Dopo che la dimensione totale dei file di registro di sistema Greengrass supera questa dimensione totale massima, il software AWS IoT Greengrass Core elimina i file di registro di sistema Greengrass più vecchi.

Questo parametro è equivalente al parametro log size limit (`totalLogsSizeKB`) del componente Greengrass nucleus. Il software AWS IoT Greengrass Core utilizza il minimo dei due valori come dimensione massima totale del registro di sistema Greengrass.

`diskSpaceLimitUnit`

(Opzionale) L'unità per. `diskSpaceLimit` Seleziona una delle opzioni seguenti:

- `KB`— kilobyte
- `MB`— megabyte
- `GB`— gigabyte

Impostazione predefinita: `KB`

`deleteLogFileAfterCloudUpload`

(Facoltativo) È possibile eliminare un file di registro dopo che il componente di gestione dei registri ha caricato i registri in Logs. CloudWatch

Impostazione predefinita: `false`

`componentLogsConfiguration`

(Facoltativo) Un elenco di configurazioni di registro per i componenti del dispositivo principale. Ogni configurazione in questo elenco definisce la configurazione del registro

per un componente o un'applicazione. Il componente log manager carica questi registri dei componenti in Logs CloudWatch

Ogni oggetto contiene le seguenti informazioni:

`componentName`

Il nome del componente o dell'applicazione per questa configurazione di registro. È possibile specificare il nome di un componente Greengrass o un altro valore per identificare questo gruppo di log.

`minimumLogLevel`

(Facoltativo) Il livello minimo di messaggi di registro da caricare. Questo livello minimo si applica solo se i log di questo componente utilizzano un formato JSON specifico, disponibile nel repository del modulo di AWS IoT Greengrass registrazione su. GitHub

Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`diskSpaceLimit`

(Facoltativo) La dimensione totale massima di tutti i file di registro per questo componente, nell'unità specificata`diskSpaceLimitUnit`. Dopo che la dimensione totale dei file di registro di questo componente supera questa dimensione totale massima, il software AWS IoT Greengrass Core elimina i file di registro più vecchi di questo componente.

Questo parametro è correlato al parametro limite di dimensione del registro (`totalLogsSizeKB`) del componente Greengrass nucleus. Il software AWS IoT Greengrass Core utilizza il minimo dei due valori come dimensione massima totale del registro per questo componente.

`diskSpaceLimitUnit`

(Facoltativo) L'unità per`diskSpaceLimit`. Seleziona una delle opzioni seguenti:

- `KB`— kilobyte
- `MB`— megabyte
- `GB`— gigabyte

Impostazione predefinita: `KB`

`logFileDirectoryPath`

(Facoltativo) Il percorso della cartella che contiene i file di registro di questo componente.

Non è necessario specificare questo parametro per i componenti Greengrass che stampano su standard output (stdout) e standard error (stderr).

Default: */greengrass/v2*`/logs`.

`logFileRegex`

(Facoltativo) Un'espressione regolare che specifica il formato del nome del file di registro utilizzato dal componente o dall'applicazione. Il componente log manager utilizza questa espressione regolare per identificare i file di registro nella cartella in`logFileDirectoryPath`.

Non è necessario specificare questo parametro per i componenti Greengrass che stampano su standard output (stdout) e standard error (stderr).

Se il componente o l'applicazione ruota i file di registro, specificate un'espressione regolare che corrisponda ai nomi dei file di registro ruotati. Ad esempio, potresti specificare di caricare i log **`hello_world\\\\w*.log`** per un'applicazione Hello World. Il `\\\\w*` modello corrisponde a zero o più caratteri di parola, inclusi caratteri alfanumerici e caratteri di sottolineatura. Questa espressione regolare corrisponde ai file di registro con e senza timestamp nel nome. In questo esempio, il gestore dei log carica i seguenti file di registro:

- `hello_world.log`— Il file di registro più recente per l'applicazione Hello World.
- `hello_world_2020_12_15_17_0.log`— Un vecchio file di registro per l'applicazione Hello World.

Impostazione predefinita:*componentName*`\\\\w*.log`, dove *componentName* è il nome del componente per questa configurazione di registro.

`deleteLogFileAfterCloudUpload`

(Facoltativo) È possibile eliminare un file di registro dopo che il componente di gestione dei registri ha caricato i log in Logs. CloudWatch

Impostazione predefinita: `false`

`multiLineStartPattern`

(Facoltativo) Un'espressione regolare che identifica quando un messaggio di registro su una nuova riga è un nuovo messaggio di registro. Se l'espressione regolare non corrisponde alla nuova riga, il componente log manager aggiunge la nuova riga al messaggio di registro della riga precedente.

Per impostazione predefinita, il componente log manager controlla se la riga inizia con uno spazio bianco, ad esempio una tabulazione o uno spazio. In caso contrario, il gestore dei registri gestisce quella riga come un nuovo messaggio di registro. Altrimenti, aggiunge quella riga al messaggio di registro corrente. Questo comportamento garantisce che il componente di gestione dei log non divida i messaggi che si estendono su più righe, come le tracce dello stack.

`periodicUploadIntervalSec`

(Facoltativo) Il periodo in secondi in cui il componente di gestione dei registri verifica la presenza di nuovi file di registro da caricare.

Impostazione predefinita: `300` (5 minuti)

Minimo: `0.000001` (1 microsecondo)

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica di caricare i registri di sistema e i registri dei `com.example.HelloWorld` componenti in Logs. CloudWatch

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true",
      "minimumLogLevel": "INFO",
      "diskSpaceLimit": "10",
```

```
      "diskSpaceLimitUnit": "MB",
      "deleteLogFileAfterCloudUpload": "false"
    },
    "componentLogsConfiguration": [
      {
        "componentName": "com.example.HelloWorld",
        "minimumLogLevel": "INFO",
        "diskSpaceLimit": "20",
        "diskSpaceLimitUnit": "MB",
        "deleteLogFileAfterCloudUpload": "false"
      }
    ]
  },
  "periodicUploadIntervalSec": "300"
}
```

v2.0.x

`logsUploaderConfiguration`

(Facoltativo) La configurazione per i log caricati dal componente Log Manager. Questo oggetto contiene le seguenti informazioni:

`systemLogsConfiguration`

(Facoltativo) La configurazione per i registri di sistema del software AWS IoT Greengrass Core. Specificate questa configurazione per consentire al componente di gestione dei log di gestire i registri di sistema. Questo oggetto contiene le seguenti informazioni:

`uploadToCloudWatch`

(Facoltativo) È possibile caricare i log di sistema in CloudWatch Logs.

Impostazione predefinita: `false`

`minimumLogLevel`

(Facoltativo) Il livello minimo di messaggi di registro da caricare. Questo livello minimo si applica solo se si configura il componente Greengrass nucleus per l'output di log in formato JSON. Per abilitare i log in formato JSON, specificate il parametro del formato JSON di registrazione (). `logging.format`

Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`diskSpaceLimit`

(Facoltativo) La dimensione totale massima dei file di registro del sistema Greengrass, nell'unità specificata. `diskSpaceLimitUnit` Dopo che la dimensione totale dei file di registro di sistema Greengrass supera questa dimensione totale massima, il software AWS IoT Greengrass Core elimina i file di registro di sistema Greengrass più vecchi.

Questo parametro è equivalente al parametro log size limit (`totalLogsSizeKB`) del componente Greengrass nucleus. Il software AWS IoT Greengrass Core utilizza il minimo dei due valori come dimensione massima totale del registro di sistema Greengrass.

`diskSpaceLimitUnit`

(Opzionale) L'unità per. `diskSpaceLimit` Seleziona una delle opzioni seguenti:

- `KB`— kilobyte
- `MB`— megabyte
- `GB`— gigabyte

Impostazione predefinita: `KB`

`deleteLogFileAfterCloudUpload`

(Facoltativo) È possibile eliminare un file di registro dopo che il componente di gestione dei registri ha caricato i registri in Logs. CloudWatch

Impostazione predefinita: `false`

`componentLogsConfiguration`

(Facoltativo) Un elenco di configurazioni di registro per i componenti del dispositivo principale. Ogni configurazione in questo elenco definisce la configurazione del registro per un componente o un'applicazione. Il componente log manager carica questi registri dei componenti in Logs CloudWatch

Ogni oggetto contiene le seguenti informazioni:

`componentName`

Il nome del componente o dell'applicazione per questa configurazione di registro. È possibile specificare il nome di un componente Greengrass o un altro valore per identificare questo gruppo di log.

`minimumLogLevel`

(Facoltativo) Il livello minimo di messaggi di registro da caricare. Questo livello minimo si applica solo se i log di questo componente utilizzano un formato JSON specifico, che puoi trovare nel repository del modulo di AWS IoT Greengrass registrazione su. GitHub

Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`diskSpaceLimit`

(Facoltativo) La dimensione totale massima di tutti i file di registro per questo componente, nell'unità specificata`diskSpaceLimitUnit`. Dopo che la dimensione totale dei file di registro di questo componente supera questa dimensione totale massima, il software AWS IoT Greengrass Core elimina i file di registro più vecchi di questo componente.

Questo parametro è correlato al parametro limite di dimensione del registro (`totalLogsSizeKB`) del componente Greengrass nucleus. Il software AWS IoT Greengrass Core utilizza il minimo dei due valori come dimensione massima totale del registro per questo componente.

`diskSpaceLimitUnit`

(Facoltativo) L'unità per`diskSpaceLimit`. Seleziona una delle opzioni seguenti:

- `KB`— kilobyte
- `MB`— megabyte

- `GB`— gigabyte

Impostazione predefinita: `KB`

`logFileDirectoryPath`

Il percorso della cartella che contiene i file di registro di questo componente.

Per caricare i log di un componente Greengrass***/greengrass/v2*****/logs**, specificate e sostituiteli con */greengrass/v2* la cartella principale di Greengrass.

`logFileRegex`

Un'espressione regolare che specifica il formato del nome del file di registro utilizzato dal componente o dall'applicazione. Il componente log manager utilizza questa espressione regolare per identificare i file di registro nella cartella in`logFileDirectoryPath`.

Per caricare i log di un componente Greengrass, specifica un'espressione regolare che corrisponda ai nomi dei file di registro ruotati. Ad esempio, potresti specificare di caricare i log **`com.example.HelloWorld\\w*.log`** per un componente Hello World. Il `\\w*` modello corrisponde a zero o più caratteri di parola, inclusi caratteri alfanumerici e caratteri di sottolineatura. Questa espressione regolare corrisponde ai file di registro con e senza timestamp nel nome. In questo esempio, il gestore dei log carica i seguenti file di registro:

- `com.example.HelloWorld.log`— Il file di registro più recente per il componente Hello World.
- `com.example.HelloWorld_2020_12_15_17_0.log`— Un vecchio file di registro per il componente Hello World. Il nucleo Greengrass aggiunge un timestamp rotante ai file di registro.

`deleteLogFileAfterCloudUpload`

(Facoltativo) È possibile eliminare un file di registro dopo che il componente di gestione dei registri ha caricato i log in Logs. CloudWatch

Impostazione predefinita: `false`

`multiLineStartPattern`

(Facoltativo) Un'espressione regolare che identifica quando un messaggio di registro su una nuova riga è un nuovo messaggio di registro. Se l'espressione regolare non

corrisponde alla nuova riga, il componente log manager aggiunge la nuova riga al messaggio di registro della riga precedente.

Per impostazione predefinita, il componente log manager controlla se la riga inizia con uno spazio bianco, ad esempio una tabulazione o uno spazio. In caso contrario, il gestore dei registri gestisce quella riga come un nuovo messaggio di registro. Altrimenti, aggiunge quella riga al messaggio di registro corrente. Questo comportamento garantisce che il componente di gestione dei log non divida i messaggi che si estendono su più righe, come le tracce dello stack.

`periodicUploadIntervalSec`

(Facoltativo) Il periodo in secondi in cui il componente di gestione dei registri verifica la presenza di nuovi file di registro da caricare.

Impostazione predefinita: `300` (5 minuti)

Minimo: `0.000001` (1 microsecondo)

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica di caricare i registri di sistema e i registri dei `com.example.HelloWorld` componenti in Logs. CloudWatch

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true",
      "minimumLogLevel": "INFO",
      "diskSpaceLimit": "10",
      "diskSpaceLimitUnit": "MB",
      "deleteLogFileAfterCloudUpload": "false"
    },
    "componentLogsConfiguration": [
      {
        "componentName": "com.example.HelloWorld",
        "minimumLogLevel": "INFO",
        "logFileDirectoryPath": "/greengrass/v2/logs",
        "logFileRegex": "com.example.HelloWorld\\w*.log",
        "diskSpaceLimit": "20",
        "diskSpaceLimitUnit": "MB",
        "deleteLogFileAfterCloudUpload": "false"
```

```
      }
    ]
  },
  "periodicUploadIntervalSec": "300"
}
```

Utilizzo

Il componente log manager viene caricato nei seguenti gruppi di log e flussi di log.

2.1.0 and later

Nome del gruppo di log

```
/aws/greengrass/componentType/region/componentName
```

Il nome del gruppo di log utilizza le seguenti variabili:

- `componentType`— Il tipo di componente, che può essere uno dei seguenti:
  - `GreengrassSystemComponent`— Questo gruppo di log include i log per i componenti del nucleo e del plugin, che vengono eseguiti nella stessa JVM del nucleo Greengrass. Il componente fa parte del nucleo Greengrass.
  - `UserComponent`— Questo gruppo di log include i registri per componenti generici, componenti Lambda e altre applicazioni sul dispositivo. Il componente non fa parte del nucleo Greengrass.

  Per ulteriori informazioni, consulta Tipi di componenti.
- `region`— La AWS regione utilizzata dal dispositivo principale.
- `componentName`— Il nome del componente. Per i registri di sistema, questo valore è`System`.

Nome del flusso di log

```
/date/thing/thingName
```

Il nome del flusso di registro utilizza le seguenti variabili:

- `date`— La data del registro, ad esempio`2020/12/15`. Il componente log manager utilizza il `yyyy/MM/dd` formato.
- `thingName`— Il nome del dispositivo principale.

> **Note**
>
> Se il nome di un oggetto contiene due punti (`:`), il gestore dei registri sostituisce i due punti con un segno più (+).

2.0.x

Nome del gruppo di log

```
/aws/greengrass/componentType/region/componentName
```

Il nome del gruppo di log utilizza le seguenti variabili:

- `componentType`— Il tipo di componente, che può essere uno dei seguenti:
  - `GreengrassSystemComponent`— Il componente fa parte del nucleo Greengrass.
  - `UserComponent`— Il componente non fa parte del nucleo Greengrass. Il gestore dei registri utilizza questo tipo per i componenti Greengrass e altre applicazioni sul dispositivo.
- `region`— La AWS regione utilizzata dal dispositivo principale.
- `componentName`— Il nome del componente. Per i registri di sistema, questo valore è`System`.

Nome del flusso di log

```
/date/deploymentTargets/thingName
```

Il nome del flusso di registro utilizza le seguenti variabili:

- `date`— La data del registro, ad esempio`2020/12/15`. Il componente log manager utilizza il `yyyy/MM/dd` formato.
- `deploymentTargets`— Le cose le cui distribuzioni includono il componente. Il componente log manager separa ogni destinazione con una barra. Se il componente viene eseguito sul dispositivo principale come risultato di una distribuzione locale, questo valore è. `LOCAL_DEPLOYMENT`

  Si consideri un esempio in cui si dispone di un dispositivo principale denominato `MyGreengrassCore` e il dispositivo principale ha due distribuzioni:

  - Una distribuzione destinata al dispositivo principale,. `MyGreengrassCore`

  - Una distribuzione destinata a un gruppo di oggetti denominato`MyGreengrassCoreGroup`, che contiene il dispositivo principale.

    I `deploymentTargets` per questo dispositivo principale sono`thing/MyGreengrassCore/thinggroup/MyGreengrassCoreGroup`.
- `thingName`— Il nome del dispositivo principale.

Formati per le voci di registro.

Il nucleo Greengrass scrive i file di registro in formato stringa o JSON. Per i log di sistema, è possibile controllare il formato impostando il `format` campo della voce. `logging` Puoi trovare la `logging` voce nel file di configurazione del componente Greengrass nucleus. Per ulteriori informazioni, vedere Greengrass nucleus configuration.

Il formato di testo è in formato libero e accetta qualsiasi stringa. Il seguente messaggio del servizio Fleet Status è un esempio di registrazione in formato stringa:

```
2023-03-26T18:18:27.271Z [INFO] (pool-1-thread-2)
com.aws.greengrass.status.FleetStatusService: fss-status-update-published.
Status update published to FSS. {trigger=CADENCE, serviceName=FleetStatusService,
currentState=RUNNING}
```

È necessario utilizzare il formato JSON se si desidera visualizzare i log con il comando Greengrass CLI logs o interagire con i log a livello di codice. L'esempio seguente delinea la forma JSON:

```
{
   "loggerName": <string>,
   "level": <"DEBUG" | "INFO" | "ERROR" | "TRACE" | "WARN">,
   "eventType": <string, optional>,
   "cause": <string, optional>,
   "contexts": {},
   "thread": <string>,
   "message": <string>,
   "timestamp": <epoch time> # Needs to be epoch time
}
```

Per controllare l'output dei log del componente, puoi utilizzare l'`minimumLogLevel`opzione di configurazione. Per utilizzare questa opzione, il componente deve scrivere le sue voci di registro in formato JSON. È necessario utilizzare lo stesso formato del file di registro di sistema.

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.3.7	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.3.6	Correzioni di bug e miglioramenti • Regola i livelli di registro per determinati errori.

Versione	Modifiche
2.3.5	Miglioramenti Migliora la velocità di caricamento dei log. Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.3.4	Correzioni di bug e miglioramenti • Aggiunge il supporto per l'impostazione del `periodicUploadIntervalSec` parametro su valori frazionari. Il valore minimo è 1 microsecondo. • Risolve un problema per cui il gestore dei registri non rispetta i CloudWatch `putLogEvents` limiti.
2.3.3	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.3.2	Correzioni di bug e miglioramenti • Migliora la gestione dello spazio in modo che i file di registro non vengano eliminati prima di essere caricati. • Risolve i problemi relativi alla gestione della cache. • Correzioni e miglioramenti aggiuntivi di bug minori.
2.3.1	Correzioni di bug e miglioramenti • Risolve un problema in cui i gruppi di file di destinazione con più file di registro attivi caricavano voci duplicate. CloudWatch • Correzioni e miglioramenti aggiuntivi di bug minori.

Versione	Modifiche
2.3.0	**Note** Si consiglia di eseguire l'aggiornamento a Greengrass nucleus 2.9.1 quando si esegue l'aggiornamento a log manager 2.3.0. Nuove funzionalità Riduce i ritardi nei log elaborando e caricando direttamente i file di registro attivi anziché attendere la rotazione dei nuovi file. Correzioni di bug e miglioramenti • Migliora il supporto della rotazione dei log durante la rotazione di file con un nome univoco. • Correzioni e miglioramenti aggiuntivi di bug minori.
2.2.8	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.2.7	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.2.6	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.2.5	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.2.4	Correzioni di bug e miglioramenti • Migliora la stabilità nella gestione di configurazioni non valide. • Correzioni e miglioramenti minori aggiuntivi.

Versione	Modifiche
2.2.3	Correzioni di bug e miglioramenti • Migliora la stabilità in alcuni scenari in cui il componente si riavvia o riscontra errori. • Risolve i problemi relativi al mancato caricamento di messaggi di registro e file di registro di grandi dimensioni in determinati scenari. • Risolve i problemi relativi al modo in cui questo componente gestisce gli aggiornamenti di ripristino della configurazione. • Risolve un problema a causa del quale un valore `null` `diskSpace Limit` di configurazione impediva la distribuzione del componente.
2.2.2	Correzioni di bug e miglioramenti • Aggiunge il supporto per i messaggi di registro di dimensioni superiori a 256 kilobyte. Il componente di gestione dei log divide questi messaggi di registro di grandi dimensioni in più messaggi con lo stesso timestamp degli eventi di registro.
2.2.1	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.2.0	Nuova caratteristica • Aggiunge il parametro `componentLogsConfigurationMap` di configurazione per supportare un formato di mappa per le configura zioni dei registri dei componenti. Ogni `componentName` oggetto nella mappa definisce la configurazione del registro per un componente o un'applicazione.
2.1.3	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.1.1	Correzioni di bug e miglioramenti • Risolve un problema per cui la configurazione del registro di sistema non veniva aggiornata in alcuni casi.

Versione	Modifiche
2.1.0	Correzioni di bug e miglioramenti • Usa i valori predefiniti per `logFileDirectoryPath` e `logFileRegex` che funzionano per i componenti Greengrass che stampano su output standard (stdout) ed errore standard (stderr). • Indirizza correttamente il traffico attraverso un proxy di rete configurato durante il caricamento dei log su Logs. CloudWatch • Gestisci correttamente i due punti (`:`) nei nomi dei flussi di log. CloudWatch I nomi dei flussi di log non supportano i due punti. • Semplifica i nomi dei flussi di log rimuovendo i nomi dei gruppi di oggetti dal flusso di log. • Rimuove un messaggio di registro degli errori che viene stampato durante il normale comportamento.
2.0.x	Versione iniziale.

Componenti per l'apprendimento automatico

AWS IoT Greengrassfornisce i seguenti componenti di machine learning che puoi distribuire su dispositivi supportati per eseguire inferenze di machine learning utilizzando modelli addestrati in Amazon SageMaker o con i tuoi modelli pre-addestrati archiviati in Amazon S3.

AWSfornisce le seguenti categorie di componenti di apprendimento automatico:

- Componente del modello: contiene modelli di apprendimento automatico come artefatti Greengrass.
- Componente Runtime: contiene lo script che installa il framework di machine learning e le sue dipendenze sul dispositivo principale Greengrass.
- Componente di inferenza: contiene il codice di inferenza e include le dipendenze dei componenti per installare il framework di machine learning e scaricare modelli di machine learning preaddestrati.

È possibile utilizzare il codice di inferenza di esempio e i modelli preaddestrati nei componenti di machine learning AWS forniti per eseguire la classificazione delle immagini e il rilevamento degli

oggetti utilizzando DLR e Lite. TensorFlow Per eseguire inferenze di machine learning personalizzate con i tuoi modelli archiviati in Amazon S3 o per utilizzare un framework di machine learning diverso, puoi utilizzare le ricette di questi componenti pubblici come modelli per creare componenti di machine learning personalizzati. Per ulteriori informazioni, consulta Personalizza i tuoi componenti di machine learning.

AWS IoT Greengrassinclude anche un componente AWS fornito per gestire l'installazione e il ciclo di vita dell'agente SageMaker Edge Manager sui dispositivi core Greengrass. Con SageMaker Edge Manager, puoi utilizzare i modelli SageMaker compilati da Amazon NEO direttamente sul tuo dispositivo principale. Per ulteriori informazioni, consulta Usa Amazon SageMaker Edge Manager sui dispositivi core Greengrass.

La tabella seguente elenca i componenti di machine learning disponibili in. AWS IoT Greengrass

> **Note**
>
> Diversi componenti AWS forniti dipendono da versioni minori specifiche del nucleo Greengrass. A causa di questa dipendenza, è necessario aggiornare questi componenti quando si aggiorna il nucleo di Greengrass a una nuova versione secondaria. Per informazioni sulle versioni specifiche del nucleo da cui dipende ogni componente, consultate l'argomento relativo ai componenti. Per ulteriori informazioni sull'aggiornamento del nucleo, vedere. Aggiornamento del software AWS IoT Greengrass Core (OTA)

Quando un componente ha un tipo di componente sia generico che Lambda, la versione corrente del componente è di tipo generico e una versione precedente del componente è di tipo Lambda.

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Lookout for Vision Edge Agent	Implementa il runtime Amazon Lookout for Vision sul dispositivo principale	Generico	Linux	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
	Greengrass, in modo da poter utilizzare la visione artificiale per trovare difetti nei prodotti industriali.			
SageMaker Edge Manager	Implementa l'agente Amazon SageMaker Edge Manager sul dispositivo principale Greengrass.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Classificazione delle immagini DLR	Componente di inferenza che utilizza l'archivio del modello di classificazione delle immagini DLR e il componente runtime DLR come dipendenze per installare DLR, scaricare modelli di classificazione delle immagini di esempio ed eseguire inferenze di classificazione delle immagini sui dispositivi supportati.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Rilevamento di oggetti DLR	Componente di inferenza che utilizza l'archivio del modello di rilevamento degli oggetti DLR e il componente runtime DLR come dipendenze per installare DLR, scaricare modelli di rilevamento di oggetti di esempio ed eseguire inferenze per il rilevamento degli oggetti sui dispositivi supportati.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Archivio modelli di classificazione delle immagini DLR	Componente del modello che contiene esempi di modelli di classificazione delle immagini ResNet -50 come artefatti Greengrass.	Generico	Linux, Windows	No
Archivio modelli DLR per il rilevamento di oggetti	Componente del modello che contiene esempi di modelli di rilevamento di oggetti YOLov3 come artefatti Greengrass.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Runtime DLR	Componente di runtime che contiene uno script di installazione utilizzato per installare DLR e le sue dipendenze sul dispositivo principale Greengrass.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
TensorFlow Classificazione delle immagini Lite	Componente di inferenza che utilizza l' TensorFlow archivio del modello di classificazione delle immagini TensorFlow Lite e il componente runtime Lite come dipendenze per installare TensorFlow Lite, scaricare modelli di classificazione delle immagini di esempio ed eseguire inferenze di classificazione delle immagini sui dispositivi supportati.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
TensorFlow Rilevamento di oggetti Lite	Componente di inferenza che utilizza l'archivio del modello di rilevamento degli oggetti TensorFlow Lite e il componente runtime TensorFlow Lite come dipendenze per installare TensorFlow Lite, scaricare modelli di rilevamento di oggetti di esempio ed eseguire inferenze per il rilevamento degli oggetti sui dispositivi supportati.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
TensorFlow Archivio di modelli di classificazione delle immagini Lite	Componente del modello che contiene un modello MobileNet v1 di esempio come artefatto Greengrass.	Generico	Linux, Windows	No
TensorFlow Archivio modelli Lite per il rilevamento di oggetti	Componente del modello che contiene un MobileNet modello di Single Shot Detection (SSD) di esempio come artefatto Greengrass.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
TensorFlow Runtime Lite	Componente di runtime che contiene uno script di installazione utilizzato per installare TensorFlow Lite e le sue dipendenze sul dispositivo principale Greengrass.	Generico	Linux, Windows	No

Lookout for Vision Edge Agent

Il componente Lookout for Vision Edge Agent `aws.iot.lookoutvision.EdgeAgent` () installa un server di runtime Amazon Lookout for Vision locale, che utilizza la visione artificiale per individuare difetti visivi nei prodotti industriali.

Per utilizzare questo componente, crea e distribuisci i componenti del modello di apprendimento automatico Lookout for Vision. Questi modelli di apprendimento automatico prevedono la presenza di anomalie nelle immagini trovando schemi nelle immagini utilizzate per addestrare il modello. Quindi, puoi sviluppare e distribuire componenti Greengrass personalizzati, denominati componenti di applicazioni client, che forniscono immagini e flussi video a questo componente di runtime per rilevare anomalie utilizzando i modelli di apprendimento automatico.

È possibile utilizzare l'API Lookout for Vision Edge Agent per interagire con questo componente di altri componenti Greengrass. Questa API è implementata utilizzando gRPC, che è un protocollo per effettuare chiamate di procedura remota. Per ulteriori informazioni, consulta Writing a client application component e il riferimento all'API Lookout for Vision Edge Agent nella Amazon Lookout for Vision Developer Guide.

Per ulteriori informazioni su come utilizzare questo componente, consulta quanto segue:

- Usa Amazon Lookout Greengrass visione.
- Cos'è Amazon Lookout for Vision? nella Guida per sviluppatori di Amazon Lookout for Vision
- Creazione di un modello Lookout for Vision nella Amazon Lookout for Vision Developer Guide.
- Utilizzo di un modello Lookout for Vision su un dispositivo edge nella Amazon Lookout for Vision Developer Guide.

> **Note**
>
> Il componente Lookout for Vision Edge Agent è disponibile solo nei seguenti Regioni AWS casi:
>
> - Stati Uniti orientali (Ohio)
> - Stati Uniti orientali (Virginia settentrionale)
> - US West (Oregon)
> - Europa (Francoforte)
> - Europa (Irlanda)
> - Asia Pacifico (Tokyo)
> - Asia Pacifico (Seul)

Argomenti

- Versioni
- Type
- Sistema operativo
- Requisiti
- Dipendenze
- Configurazione
- File di registro locale
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 1.2.x
- 1.1.x
- 1.0.x
- 0,1.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato solo sui dispositivi principali di Linux.

Requisiti

Questo componente ha i seguenti requisiti:

- Il dispositivo principale Greengrass deve utilizzare un'architettura Armv8 (AArch64) o x86_64.
- Se si utilizza la versione 1.0.0 o successiva di questo componente, Python 3.8 o Python 3.9, incluso, `pip` installato sul dispositivo principale Greengrass.

  Se si utilizza la versione 0.1.x di questo componente, Python 3.7, incluso, `pip` installato sul dispositivo principale Greengrass.

  > **Important**
  >
  > Il dispositivo deve avere una di queste versioni esatte di Python. Questo componente non supporta le versioni successive di Python.

- Per utilizzare l'inferenza delle unità di elaborazione grafica (GPU), il dispositivo principale deve soddisfare i seguenti requisiti. L'inferenza tramite GPU è facoltativa nella versione 1.1.0 e successive di questo componente.

- Un'unità di elaborazione grafica (GPU) che supporta CUDA. Per ulteriori informazioni, consulta Verifica di disporre di una GPU compatibile con CUDA nella documentazione CUDA Toolkit.
- cuDNN, CUDA e TensorRT installati sul dispositivo principale Greengrass.
  - Sui dispositivi NVIDIA Jetson, come Jetson Nano o Jetson Xavier, cuDNN, CUDA e TensorRT vengono installati con NVIDIA. JetPack Non è necessario apportare alcuna modifica. Questo componente supporta JetPack 4.4, JetPack4.5, JetPack 4.5.1 e JetPack4.6.1.

    > **Important**
    >
    > È necessario installare una di queste versioni JetPack e non un'altra versione. Il servizio Lookout for Vision compila modelli di visione artificiale per JetPack queste piattaforme.

  - Sui dispositivi x86 con una GPU dotata della microarchitettura NVIDIA Ampere (o la capacità di elaborazione della GPU è 8.0), procedi come segue:
    - Installa cuDNN seguendo le istruzioni nella Guida all'installazione di NVIDIA cuDNN.
    - Installa CUDA versione 11.2 seguendo le istruzioni nella Guida all'installazione di NVIDIA CUDA per Linux.
    - Installa TensorRT versione 8.2.0 seguendo le istruzioni nella documentazione NVIDIA TensorRT.
  - Sui dispositivi x86 con una GPU con un'architettura NVIDIA precedente ad Ampere (o la capacità di calcolo della GPU è inferiore a 8.0), procedi come segue:
    - Installa cuDNN seguendo le istruzioni nella Guida all'installazione di NVIDIA cuDNN.
    - Installa CUDA versione 10.2 seguendo le istruzioni nella Guida all'installazione di NVIDIA CUDA per Linux.
    - Installa TensorRT versione 7.1.3 o successiva, ma precedente alla versione 8.0.0, seguendo le istruzioni nella documentazione di NVIDIA TensorRT.
- L'utente di sistema che esegue questo componente deve essere un membro del gruppo di sistema che ha accesso alla GPU del dispositivo. Il nome di questo gruppo varia in base al sistema operativo. Consulta la documentazione del tuo sistema operativo e della GPU per determinare il nome di questo gruppo di sistemi.

  Ad esempio, sui dispositivi NVIDIA Jetson, il nome di questo gruppo è ed è `video` possibile eseguire il comando seguente per aggiungere un utente di sistema a questo gruppo. Sostituisci *`ggc_user con il nome dell'utente`* da aggiungere.

```
sudo usermod -aG video ggc_user
```

Dipendenze

Questo componente non ha alcuna dipendenza.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare quando si distribuisce il componente.

`Socket`

(Facoltativo) Il file socket in cui opera l'Edge Agent. I componenti del modello Lookout for Vision utilizzano questo file socket per comunicare con Edge Agent. Se si modifica questo parametro, è necessario specificare lo stesso valore quando si distribuiscono i componenti del modello Lookout for Vision.

Impostazione predefinita: `unix:///tmp/aws.iot.lookoutvision.EdgeAgent.sock`

File di registro locale

Questo componente utilizza il seguente file di registro.

```
/greengrass/v2/logs/aws.iot.lookoutvision.EdgeAgent.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale.

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.lookoutvision.EdgeAgent.log
  ```

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
1.2.0	Correzioni di bug generali e miglioramenti.
1.1.9	Correzioni di bug generali e miglioramenti.
1.1.8	Correzioni di bug generali e miglioramenti.
1.1.7	Nuove funzionalità • Installa il `opencv-python-headless` pacchetto nell'ambiente virtuale Lookout for Vision Edge Agent. Correzioni di bug e miglioramenti • Migliora il calcolo del punteggio di confidenza. • Ridimensiona la maschera del modello della mappa termica alla dimensione originale del file. • Correzioni di bug generali e miglioramenti.
1.1.6	Nuove funzionalità Aggiunti nuovi valori al risultato. `DetectAnomalies` • `anomaly_score` — Il numero compreso tra 0,0 e 1,0 che indica quanto sia anomala un'immagine. • `anomaly_threshold` — Soglia impostata durante l'addestramento del modello che determina il confine tra un'immagine anomala e un'immagi ne normale. Correzioni di bug generali e miglioramenti.
1.1.4	Nuove funzionalità Aggiunto il supporto per OpenCV per il ridimensionamento delle immagini quando disponibile. L'agente Edge utilizza Pillow quando OpenCV non è disponibile. Correzioni di bug e miglioramenti Correzioni di bug generali e miglioramenti.

Versione	Modifiche
1.1.3	Correzioni di bug generali e miglioramenti.
1.1.1	Correzioni di bug generali e miglioramenti.
1.1.0	Nuove funzionalità • Aggiunge il supporto per i modelli di segmentazione delle immagini, che identificano le anomalie nelle immagini. • Aggiunge il supporto per l'inferenza della CPU, in modo da poter utilizzar e i modelli Lookout for Vision su dispositivi core senza GPU. Correzioni di bug e miglioramenti • Correzioni di bug generali e miglioramenti.
1.0.0	Questa versione del componente Lookout for Vision Edge Agent richiede una versione di Python diversa dalla versione 0.1.x. Se desideri eseguire l'aggiorn amento dalla v0.1.x alla v1.x, devi aggiornare l'installazione di Python sul dispositivo principale. Correzioni di bug e miglioramenti • Correzioni di bug generali e miglioramenti.
0.1.37	Correzioni di bug generali e miglioramenti.
0,1,36	Versione iniziale.

SageMaker Edge Manager

> **Important**
>
> SageMaker Edge Manager verrà interrotto il 26 aprile 2024. Per ulteriori informazioni su come continuare a distribuire i modelli sui dispositivi edge, consulta SageMaker Edge Manager End of Life.

Il componente Amazon SageMaker Edge Manager (`aws.greengrass.SageMakerEdgeManager`) installa il binario dell'agente SageMaker Edge Manager.

SageMaker Edge Manager fornisce la gestione dei modelli per i dispositivi edge in modo da ottimizzare, proteggere, monitorare e mantenere i modelli di machine learning su flotte di dispositivi edge. Il componente SageMaker Edge Manager installa e gestisce il ciclo di vita dell'agente SageMaker Edge Manager sul dispositivo principale. È inoltre possibile utilizzare SageMaker Edge Manager per impacchettare e utilizzare modelli SageMaker NEO-compilati come componenti del modello sui dispositivi core Greengrass. Per ulteriori informazioni sull'utilizzo dell'agente SageMaker Edge Manager sul dispositivo principale, vedere. Usa Amazon SageMaker Edge Manager sui dispositivi core Greengrass

SageMaker Il componente Edge Manager v1.3.x installa il binario dell'agente Edge Manager v1.20220822.836f3023. Per ulteriori informazioni sulle versioni binarie dell'agente Edge Manager, vedere Edge Manager Agent.

> **Note**
>
> Il componente SageMaker Edge Manager è disponibile solo nei seguenti casiRegioni AWS:
>
> - Stati Uniti orientali (Ohio)
> - Stati Uniti orientali (Virginia settentrionale)
> - US West (Oregon)
> - UE (Francoforte)
> - UE (Irlanda)
> - Asia Pacifico (Tokyo)

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 1.3.x
- 1.2.x
- 1.1.x
- 1.0.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Un dispositivo core Greengrass in esecuzione su Amazon Linux 2, una piattaforma Linux basata su Debian (x86_64 o Armv8) o Windows (x86_64). Se non lo hai, consultare Tutorial: Nozioni di base su AWS IoT Greengrass V2.
- Python 3.6 o versione successiva, inclusa `pip` la tua versione di Python, installato sul tuo dispositivo principale.
- Il ruolo del dispositivo Greengrass è configurato con quanto segue:
  - Una relazione di fiducia che consente `credentials.iot.amazonaws.com` e consente `sagemaker.amazonaws.com` di assumere il ruolo, come illustrato nel seguente esempio di policy IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.iot.amazonaws.com"
       },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- La politica gestita da AmazonSageMakerEdgeDeviceFleetPolicyIAM.
- L'`s3:PutObject`azione, come illustrato nel seguente esempio di policy IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

- Un bucket Amazon S3 creato nello stesso Regione AWS dispositivo Account AWS principale Greengrass. SageMaker Edge Manager richiede un bucket S3 per creare una flotta di dispositivi

edge e per archiviare dati di esempio derivanti dall'esecuzione dell'inferenza sul dispositivo. Per informazioni sulla creazione di bucket S3, consulta Guida introduttiva ad Amazon S3.

- Una flotta di dispositivi SageMaker edge che utilizza lo stesso alias di AWS IoT ruolo del dispositivo principale Greengrass. Per ulteriori informazioni, consulta Crea una flotta di dispositivi edge.
- Il tuo dispositivo principale Greengrass è stato registrato come dispositivo edge nel tuo parco dispositivi SageMaker Edge. Il nome del dispositivo edge deve corrispondere al nome dell'AWS IoToggetto del dispositivo principale. Per ulteriori informazioni, consulta Registra il tuo dispositivo Greengrass core.

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`edge.sagemaker.`*`region`*`.amazonaws.com`	443	Sì	Controlla lo stato di registrazione del dispositivo e invia le metriche a. SageMaker
`*.s3.amazonaws.com`	443	Sì	Carica i dati di acquisizione nel bucket S3 che hai specificato. Puoi sostituirlo *

Endpoint	Porta	Richiesto	Descrizione
			con il nome di ogni bucket in cui carichi i dati.

Dipendenze

Quando distribuisci un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle sue dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

1.3.5

La tabella seguente elenca le dipendenze per la versione 1.3.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

1.3.4

La tabella seguente elenca le dipendenze per la versione 1.3.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

1.3.3

La tabella seguente elenca le dipendenze per la versione 1.3.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

1.3.2

La tabella seguente elenca le dipendenze per la versione 1.3.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

1.3.1

La tabella seguente elenca le dipendenze per la versione 1.3.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

1.1.1 - 1.3.0

La tabella seguente elenca le dipendenze per le versioni 1.1.1 - 1.3.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili

Dipendenza	Versioni compatibili	Tipo di dipendenza
Servizio di scambio di token	>=0.0.0	Rigidi

1.1.0

La tabella seguente elenca le dipendenze per la versione 1.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

1.0.3

La tabella seguente elenca le dipendenze per la versione 1.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

1.0.1 and 1.0.2

La tabella seguente elenca le dipendenze per le versioni 1.0.1 e 1.0.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

1.0.0

La tabella seguente elenca le dipendenze per la versione 1.0.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

> **Note**
>
> Questa sezione descrive i parametri di configurazione impostati nel componente. Per ulteriori informazioni sulla configurazione di SageMaker Edge Manager corrispondente, consulta Edge Manager Agent nella Amazon SageMaker Developer Guide.

`DeviceFleetName`

Il nome del parco dispositivi SageMaker Edge Manager che contiene il dispositivo principale Greengrass.

È necessario specificare un valore per questo parametro nell'aggiornamento della configurazione quando si distribuisce questo componente.

`BucketName`

Il nome del bucket S3 in cui carichi i dati di inferenza acquisiti. Il nome del bucket deve contenere la stringa. `sagemaker`

Se si imposta su `CaptureDataDestinationCloud`, o se si imposta su `CaptureDataPeriodicUploadtrue`, è necessario specificare un valore per questo parametro nell'aggiornamento della configurazione quando si distribuisce questo componente.

> **Note**
>
> L'acquisizione dei dati è una SageMaker funzionalità che utilizzi per caricare input di inferenza, risultati di inferenza e dati di inferenza aggiuntivi su un bucket S3 o una directory locale per analisi future. Per ulteriori informazioni sull'utilizzo dei dati di acquisizione con SageMaker Edge Manager, consulta Manage Model nella Amazon SageMaker Developer Guide.

`CaptureDataBatchSize`

(Facoltativo) La dimensione di un batch di richieste di dati di acquisizione gestite dall'agente. Questo valore deve essere inferiore alla dimensione del buffer specificata. `CaptureDataBufferSize` Si consiglia di non superare la metà della dimensione del buffer.

L'agente gestisce un batch di richieste quando il numero di richieste nel buffer raggiunge il `CaptureDataBatchSize` numero o allo scadere dell'`CaptureDataPushPeriodSeconds`intervallo, a seconda dell'evento che si verifica per primo.

Impostazione predefinita: `10`

`CaptureDataBufferSize`

(Facoltativo) Il numero massimo di richieste di dati di acquisizione archiviate nel buffer.

Impostazione predefinita: `30`

`CaptureDataDestination`

(Facoltativo) La destinazione in cui vengono archiviati i dati acquisiti. Questo parametro può avere i seguenti valori:

- `Cloud`—Carica i dati acquisiti nel bucket S3 specificato in. `BucketName`
- `Disk`—Scrive i dati acquisiti nella directory di lavoro del componente.

Se lo specifichi`Disk`, puoi anche scegliere di caricare periodicamente i dati acquisiti nel tuo bucket S3 impostando su. `CaptureDataPeriodicUpload` `true`

Impostazione predefinita: `Cloud`

`CaptureDataPeriodicUpload`

(Facoltativo) Valore di stringa che specifica se caricare periodicamente i dati acquisiti. I valori supportati sono `true` e `false`.

Imposta questo parametro su `true` se lo hai `CaptureDataDestination` impostato `Disk` e desideri inoltre che l'agente carichi periodicamente i dati acquisiti nel tuo bucket S3.

Impostazione predefinita: `false`

`CaptureDataPeriodicUploadPeriodSeconds`

(Facoltativo) L'intervallo in secondi con cui l'agente SageMaker Edge Manager carica i dati acquisiti nel bucket S3. Utilizzate questo parametro se lo impostate su. `CaptureDataPeriodicUpload` `true`

Impostazione predefinita: 8

`CaptureDataPushPeriodSeconds`

(Facoltativo) L'intervallo in secondi in cui l'agente SageMaker Edge Manager gestisce un batch di richieste di acquisizione di dati dal buffer.

L'agente gestisce un batch di richieste quando il numero di richieste nel buffer raggiunge il `CaptureDataBatchSize` numero o allo scadere dell'`CaptureDataPushPeriodSeconds`intervallo, a seconda dell'evento che si verifica per primo.

Impostazione predefinita: `4`

`CaptureDataBase64EmbedLimit`

(Facoltativo) La dimensione massima in byte dei dati acquisiti caricati dall'agente Edge Manager. SageMaker

Impostazione predefinita: `3072`

`FolderPrefix`

(Facoltativo) Il nome della cartella in cui l'agente scrive i dati acquisiti. Se `CaptureDataDestination` si imposta su`Disk`, l'agente crea la cartella nella directory specificata da`CaptureDataDiskPath`. Se lo `CaptureDataDestination` imposti su`Cloud`, o se lo `CaptureDataPeriodicUpload` impostitrue, l'agente crea la cartella nel tuo bucket S3.

Impostazione predefinita: `sme-capture`

`CaptureDataDiskPath`

Questa funzionalità è disponibile nella versione 1.1.0 e successive del componente Edge Manager. SageMaker

(Facoltativo) Il percorso della cartella in cui l'agente crea la cartella dei dati acquisiti. Se si imposta su `CaptureDataDestinationDisk`, l'agente crea la cartella dei dati acquisiti in questa directory. Se non specificate questo valore, l'agente crea la cartella dei dati acquisiti nella directory di lavoro del componente. Utilizzate il `FolderPrefix` parametro per specificare il nome della cartella dei dati acquisiti.

Impostazione predefinita: */greengrass/v2*`/work/aws.greengrass.SageMakerEdgeManager/capture`

`LocalDataRootPath`

Questa funzionalità è disponibile nella versione 1.2.0 e nelle versioni successive del componente SageMaker Edge Manager.

(Facoltativo) Il percorso in cui questo componente memorizza i seguenti dati sul dispositivo principale:

- Il database locale per i dati di runtime se `DbEnable` impostato su`true`.
- SageMaker Modelli neo-compilati che questo componente scarica automaticamente quando lo `DeploymentEnable` imposti. `true`

Impostazione predefinita: */greengrass/v2*`/work/aws.greengrass.SageMakerEdgeManager`

`DbEnable`

(Facoltativo) È possibile consentire a questo componente di memorizzare i dati di runtime in un database locale per conservare i dati, nel caso in cui il componente si guasti o il dispositivo perda alimentazione.

Questo database richiede 5 MB di spazio di archiviazione sul file system del dispositivo principale.

Impostazione predefinita: `false`

`DeploymentEnable`

Questa funzionalità è disponibile nella versione 1.2.0 e nelle versioni successive del componente SageMaker Edge Manager.

(Facoltativo) Puoi abilitare questo componente per recuperare automaticamente i modelli SageMaker NEO-compilati dai quali carichi su Amazon S3. Dopo aver caricato un nuovo modello su Amazon S3, usa SageMaker Studio o l' SageMaker API per distribuire il nuovo modello su questo dispositivo principale. Quando abiliti questa funzionalità, puoi distribuire nuovi modelli sui dispositivi principali senza dover creare una distribuzione. AWS IoT Greengrass

> **Important**
>
> Per utilizzare questa funzionalità, è necessario impostare su`DbEnable`. `true` Questa funzionalità utilizza il database locale per tenere traccia dei modelli recuperati da. Cloud AWS

Impostazione predefinita: `false`

`DeploymentPollInterval`

Questa funzionalità è disponibile nella versione 1.2.0 e nelle versioni successive del componente SageMaker Edge Manager.

(Facoltativo) La quantità di tempo (in minuti) tra cui questo componente verifica la presenza di nuovi modelli da scaricare. Questa opzione si applica quando è impostata `DeploymentEnable` su`true`.

Impostazione predefinita: `1440` (1 giorno)

`DLRBackendOptions`

Questa funzionalità è disponibile nella versione 1.2.0 e successive del componente SageMaker Edge Manager.

(Facoltativo) I flag di runtime DLR da impostare nel runtime DLR utilizzato da questo componente. È possibile impostare il seguente flag:

- `TVM_TENSORRT_CACHE_DIR`— Abilita la memorizzazione nella cache del modello TensorRT. Specificate un percorso assoluto per una cartella esistente con autorizzazioni di lettura/scrittura.
- `TVM_TENSORRT_CACHE_DISK_SIZE_MB`— Assegna il limite superiore della cartella cache del modello TensorRT. Quando la dimensione della directory supera questo limite, i motori memorizzati nella cache che vengono utilizzati meno vengono eliminati. Il valore predefinito è 512 MB.

Ad esempio, puoi impostare questo parametro sul valore seguente per abilitare la memorizzazione nella cache del modello TensorRT e limitare la dimensione della cache a 800 MB.

```
TVM_TENSORRT_CACHE_DIR=/data/secured_folder/trt/cache;
 TVM_TENSORRT_CACHE_DISK_SIZE_MB=800
```

`SagemakerEdgeLogVerbose`

(Facoltativo) Valore di stringa che specifica se abilitare la registrazione di debug. I valori supportati sono `true` e `false`.

Impostazione predefinita: `false`

`UnixSocketName`

(Facoltativo) La posizione del descrittore del file socket di SageMaker Edge Manager sul dispositivo principale.

Impostazione predefinita: `/tmp/aws.greengrass.SageMakerEdgeManager.sock`

Example Esempio: fusione e aggiornamento della configurazione

La seguente configurazione di esempio specifica che il dispositivo principale fa parte di *`MyEdgeDeviceFleet`*e che l'agente scrive i dati di acquisizione sia sul dispositivo che su un bucket S3. Questa configurazione consente anche la registrazione dei debug.

```
{
    "DeviceFleetName": "MyEdgeDeviceFleet",
    "BucketName": "DOC-EXAMPLE-BUCKET",
    "CaptureDataDestination": "Disk",
    "CaptureDataPeriodicUpload": "true",
    "SagemakerEdgeLogVerbose": "true"
}
```

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.SageMakerEdgeManager.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.SageMakerEdgeManager.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.greengrass.SageMakerEdgeManager.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.SageMakerEdgeManager.log -Tail
   10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
1.3.5	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
1.3.4	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
1.3.3	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
1.3.2	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
1.3.1	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.

Versione	Modifiche
1.3.0	Nuove funzionalità • Aggiunge il supporto per la gestione delle dimensioni del disco della cache TensorRT. • Aggiunge il `TVM_TENSORRT_CACHE_DISK_SIZE_MB` flag opzionale al BackendOptions parametro DLR per impostare il limite di dimensione per i modelli memorizzati nella cache su disco. Miglioramenti • Fornisce una migliore concorrenza nelle previsioni. Ciò consente di utilizzare meglio i motori di accelerazione dei dispositivi, come le GPU.
1.2.0	Nuove funzionalità • Aggiunge il supporto per questo componente per recuperare automatic amente i modelli SageMaker NEO-compilati che carichi su Amazon S3. Quando abiliti questa funzionalità, puoi distribuire nuovi modelli sui dispositivi principali senza dover creare una distribuzione. AWS IoT Greengrass • Aggiunge il supporto per un database di backup utilizzato da questo componente per conservare i dati di runtime, in caso di guasto del componente o di interruzione dell'alimentazione del dispositivo. • Aggiunge il supporto per la configurazione dei flag di runtime DLR quando si configura questo componente.
1.1.1	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
1.1.0	Nuove funzionalità • Aggiunge il supporto per i dispositivi core Greengrass che eseguono Amazon Linux 2. • Aggiunge il nuovo parametro `CaptureDataDiskPath` di configura zione. È possibile utilizzare questo parametro per specificare il percorso della cartella dei dati acquisiti sul dispositivo. Correzioni di bug e miglioramenti • Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.

Versione	Modifiche
1.0.3	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
1.0.2	Correzioni di bug e miglioramenti Aggiorna lo script di installazione nel ciclo di vita del componente. I tuoi dispositivi principali devono ora avere Python 3.6 o versione successiva, inclusa `pip` la tua versione di Python, installato sul dispositivo prima di distribuire questo componente.
1.0.1	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
1.0.0	Versione iniziale.

Classificazione delle immagini DLR

Il componente di classificazione delle immagini DLR (`aws.greengrass.DLRImageClassification`) contiene un codice di inferenza di esempio per eseguire l'inferenza di classificazione delle immagini utilizzando i modelli Deep Learning Runtime e resnet-50. Questo componente utilizza la variante Archivio modelli di classificazione delle immagini DLR e i Runtime DLR componenti come dipendenze per scaricare DLR e i modelli di esempio.

Per utilizzare questo componente di inferenza con un modello DLR personalizzato, create una versione personalizzata del componente dipendente del Model Store. Per utilizzare il proprio codice di inferenza personalizzato, è possibile utilizzare la ricetta di questo componente come modello per creare un componente di inferenza personalizzato.

Argomenti

- Versioni
- Type
- Sistema operativo
- Requisiti
- Dipendenze
- Configurazione
- File di registro locale

- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x
- 2,0x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:

- NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

  ```
  pip3 install --upgrade numpy
  ```

- Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

  Per abilitare lo stack di telecamere precedente

  1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

     ```
     sudo raspi-config
     ```

  2. Seleziona Opzioni di interfaccia.
  3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
  4. Riavvia il dispositivo Raspberry Pi.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.13

La tabella seguente elenca le dipendenze per la versione 2.1.13 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili

Dipendenza	Versioni compatibili	Tipo di dipendenza
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.12

La tabella seguente elenca le dipendenze per la versione 2.1.12 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.11

La tabella seguente elenca le dipendenze per la versione 2.1.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.10

La tabella seguente elenca le dipendenze per la versione 2.1.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.9

La tabella seguente elenca le dipendenze per la versione 2.1.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.8

La tabella seguente elenca le dipendenze per la versione 2.1.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.4 - 2.1.5

La tabella seguente elenca le dipendenze per le versioni da 2.1.4 a 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili
Archivio di modelli di classificazione delle immagini DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.0.x

La tabella seguente elenca le dipendenze per la versione 2.0.x di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	~2.0.0	Flessibili
Archivio di modelli di classific azione delle immagini DLR	~2.0.0	Rigidi
DLR	~1,3,0	Flessibili

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare quando si distribuisce il componente.

2.1.x

`accessControl`

(Facoltativo) L'oggetto che contiene la politica di autorizzazione che consente al componente di pubblicare messaggi nell'argomento delle notifiche predefinito.

Impostazione predefinita:

```
{
   "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.DLRImageClassification:mqttproxy:1": {
         "policyDescription": "Allows access to publish via topic ml/dlr/image-
classification.",
         "operations": [
            "aws.greengrass#PublishToIoTCore"
         ],
         "resources": [
            "ml/dlr/image-classification"
         ]
      }
   }
}
```

`PublishResultsOnTopic`

(Facoltativo) L'argomento su cui si desidera pubblicare i risultati dell'inferenza. Se modificate questo valore, dovete modificare anche il valore del `accessControl` parametro `resources` in modo che corrisponda al nome dell'argomento personalizzato.

Impostazione predefinita: `ml/dlr/image-classification`

`Accelerator`

L'acceleratore che desideri utilizzare. I valori supportati sono `cpu` e `gpu`.

I modelli di esempio nel componente del modello dipendente supportano solo l'accelerazione della CPU. Per utilizzare l'accelerazione GPU con un modello personalizzato diverso, create un componente del modello personalizzato per sovrascrivere il componente del modello pubblico.

Impostazione predefinita: `cpu`

`ImageDirectory`

(Facoltativo) Il percorso della cartella sul dispositivo in cui i componenti di inferenza leggono le immagini. È possibile modificare questo valore in qualsiasi posizione del dispositivo a cui si dispone di accesso in lettura/scrittura.

Impostazione predefinita: *`/greengrass/v2`*`/packages/artifacts-unarchived/`*`component-name`*`/image_classification/sample_images/`

> **Note**
>
> Se impostate il valore di `UseCamera` to`true`, questo parametro di configurazione viene ignorato.

`ImageName`

(Facoltativo) Il nome dell'immagine che il componente di inferenza utilizza come input per effettuare una previsione. Il componente cerca l'immagine nella cartella specificata in. `ImageDirectory` Per impostazione predefinita, il componente utilizza l'immagine di esempio nella directory delle immagini predefinita. AWS IoT Greengrasssupporta i seguenti formati di immagine: `jpeg`jpg,png, enpy.

Impostazione predefinita: `cat.jpeg`

> **Note**
>
> Se si imposta il valore di `UseCamera` to`true`, questo parametro di configurazione viene ignorato.

`InferenceInterval`

(Facoltativo) Il tempo in secondi tra ogni previsione effettuata dal codice di inferenza. Il codice di inferenza di esempio viene eseguito all'infinito e ripete le previsioni all'intervallo di tempo specificato. Ad esempio, è possibile impostare un intervallo più breve se si desidera utilizzare le immagini scattate da una fotocamera per la previsione in tempo reale.

Impostazione predefinita: `3600`

`ModelResourceKey`

(Facoltativo) I modelli utilizzati nel componente del modello pubblico dipendente. Modificate questo parametro solo se sostituite il componente del modello pubblico con un componente personalizzato.

Impostazione predefinita:

```
{
    "armv7l": "DLR-resnet50-armv7l-cpu-ImageClassification",
    "aarch64": "DLR-resnet50-aarch64-cpu-ImageClassification",
    "x86_64": "DLR-resnet50-x86_64-cpu-ImageClassification",
    "windows": "DLR-resnet50-win-cpu-ImageClassification"
}
```

`UseCamera`

(Facoltativo) Valore di stringa che definisce se utilizzare le immagini di una fotocamera collegata al dispositivo principale Greengrass. I valori supportati sono `true` e `false`.

Quando impostate questo valore su`true`, il codice di inferenza di esempio accede alla fotocamera del dispositivo ed esegue l'inferenza localmente sull'immagine acquisita. I valori dei `ImageDirectory` parametri `ImageName` and vengono ignorati. Assicuratevi che l'utente

che esegue questo componente abbia accesso in lettura/scrittura alla posizione in cui la fotocamera memorizza le immagini acquisite.

Impostazione predefinita: `false`

> **Note**
>
> Quando si visualizza la ricetta di questo componente, il parametro di `UseCamera` configurazione non viene visualizzato nella configurazione predefinita. Tuttavia, è possibile modificare il valore di questo parametro in un aggiornamento di fusione della configurazione quando si distribuisce il componente.
> Se impostate su `UseCameratrue`, dovete anche creare un collegamento simbolico per consentire al componente di inferenza di accedere alla telecamera dall'ambiente virtuale creato dal componente runtime. Per ulteriori informazioni sull'utilizzo di una telecamera con i componenti di inferenza di esempio, vedere. Aggiornamento delle configurazioni dei componenti

2.0.x

`MLRootPath`

(Facoltativo) Il percorso della cartella sui dispositivi principali Linux in cui i componenti di inferenza leggono le immagini e scrivono i risultati dell'inferenza. È possibile modificare questo valore in qualsiasi posizione del dispositivo a cui l'utente che esegue questo componente ha accesso in lettura/scrittura.

Impostazione predefinita: */greengrass/v2*`/work/variant.DLR/greengrass_ml`

Impostazione predefinita: */greengrass/v2*`/work/variant.TensorFlowLite/greengrass_ml`

`Accelerator`

L'acceleratore che desideri utilizzare. I valori supportati sono `cpu` e `gpu`.

I modelli di esempio nel componente del modello dipendente supportano solo l'accelerazione della CPU. Per utilizzare l'accelerazione GPU con un modello personalizzato diverso, create un componente del modello personalizzato per sovrascrivere il componente del modello pubblico.

Impostazione predefinita: `cpu`

`ImageName`

(Facoltativo) Il nome dell'immagine che il componente di inferenza utilizza come input per effettuare una previsione. Il componente cerca l'immagine nella cartella specificata in. `ImageDirectory` La posizione predefinita è*`MLRootPath`*`/images`. AWS IoT Greengrasssupporta i seguenti formati di immagine: `jpegjpg`,`png`, `enpy`.

Impostazione predefinita: `cat.jpeg`

`InferenceInterval`

(Facoltativo) Il tempo in secondi tra ogni previsione effettuata dal codice di inferenza. Il codice di inferenza di esempio viene eseguito all'infinito e ripete le previsioni all'intervallo di tempo specificato. Ad esempio, è possibile impostare un intervallo più breve se si desidera utilizzare le immagini scattate da una fotocamera per la previsione in tempo reale.

Impostazione predefinita: `3600`

`ModelResourceKey`

(Facoltativo) I modelli utilizzati nel componente del modello pubblico dipendente. Modificate questo parametro solo se sostituite il componente del modello pubblico con un componente personalizzato.

Impostazione predefinita:

```
armv7l: "DLR-resnet50-armv7l-cpu-ImageClassification"
x86_64: "DLR-resnet50-x86_64-cpu-ImageClassification"
```

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.DLRImageClassification.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.DLRImageClassification.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.greengrass.DLRImageClassification.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.DLRImageClassification.log -
  Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.1.13	Versione aggiornata per Greengrass nucleus versione 2.12.0.
2.1.12	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.11	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.10	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.9	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.8	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.7	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.1.5	Componente rilasciato in tutto. Regioni AWS
2.1.4	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.

Versione	Modifiche
	Questa versione non è disponibile in Europa (Londra) (). `eu-west-2`
2.1.3	Versione aggiornata per Greengrass nucleus versione 2.3.0.
2.1.2	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.1.1	Nuove funzionalità • Usa Deep Learning Runtime v1.6.0. • Aggiungi il supporto per la classificazione delle immagini di esempio sulle piattaforme Armv8 (AArch64). Ciò estende il supporto dell'appr endimento automatico per i dispositivi core Greengrass che eseguono NVIDIA Jetson, come Jetson Nano. • Abilita l'integrazione della fotocamera per l'inferenza dei campioni. Utilizzate il nuovo parametro di `UseCamera` configurazione per consentire al codice di inferenza di esempio di accedere alla telecamera sul dispositivo principale Greengrass ed eseguire l'inferenza localmente sull'immagine acquisita. • Aggiungi il supporto per la pubblicazione dei risultati di inferenza a. Cloud AWS Utilizzate il nuovo parametro di `PublishResultsOnTo pic` configurazione per specificare l'argomento su cui desiderate pubblicare i risultati. • Aggiungete il nuovo parametro di `ImageDirectory` configurazione che consente di specificare una directory personalizzata per l'immagine su cui desiderate eseguire l'inferenza. Correzioni di bug e miglioramenti • Scrivi i risultati dell'inferenza nel file di registro del componente anziché in un file di inferenza separato. • Utilizzate il modulo di registrazione del software AWS IoT Greengrass Core per registrare l'output dei componenti. • Utilizzate il SDK per dispositivi AWS IoT per leggere la configurazione del componente e applicare le modifiche alla configurazione.
2.0.4	Versione iniziale.

Rilevamento di oggetti DLR

Il componente di rilevamento degli oggetti DLR (`aws.greengrass.DLRObjectDetection`) contiene un codice di inferenza di esempio per eseguire l'inferenza del rilevamento di oggetti utilizzando Deep Learning Runtime e modelli preaddestrati di esempio. Questo componente utilizza la variante Archivio modelli DLR per il rilevamento di oggetti e i Runtime DLR componenti come dipendenze per scaricare DLR e i modelli di esempio.

Per utilizzare questo componente di inferenza con un modello DLR personalizzato, create una versione personalizzata del componente dipendente del Model Store. Per utilizzare il proprio codice di inferenza personalizzato, è possibile utilizzare la ricetta di questo componente come modello per creare un componente di inferenza personalizzato.

Argomenti

- Versioni
- Type
- Sistema operativo
- Requisiti
- Dipendenze
- Configurazione
- File di registro locale
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x
- 2,0x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Raspberry Pi OS Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

    Per abilitare lo stack di telecamere precedente

    1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleziona Opzioni di interfaccia.

3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
4. Riavvia il dispositivo Raspberry Pi.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.13

La tabella seguente elenca le dipendenze per la versione 2.1.13 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.12

La tabella seguente elenca le dipendenze per la versione 2.1.12 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.11

La tabella seguente elenca le dipendenze per la versione 2.1.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.10

La tabella seguente elenca le dipendenze per la versione 2.1.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.9

La tabella seguente elenca le dipendenze per la versione 2.1.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.8

La tabella seguente elenca le dipendenze per la versione 2.1.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.4 - 2.1.5

La tabella seguente elenca le dipendenze per le versioni da 2.1.4 a 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.1.0	Rigidi
DLR	~1.6.0	Rigidi

2.0.x

La tabella seguente elenca le dipendenze per la versione 2.0.x di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	~2.0.0	Flessibili
Archivio di modelli di rilevamento di oggetti DLR	~2.0.0	Rigidi
DLR	~1,3,0	Flessibili

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare quando si distribuisce il componente.

2.1.x

`accessControl`

(Facoltativo) L'oggetto che contiene la politica di autorizzazione che consente al componente di pubblicare messaggi nell'argomento delle notifiche predefinito.

Impostazione predefinita:

```
{
   "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.DLRObjectDetection:mqttproxy:1": {
         "policyDescription": "Allows access to publish via topic ml/dlr/object-
detection.",
         "operations": [
            "aws.greengrass#PublishToIoTCore"
         ],
         "resources": [
            "ml/dlr/object-detection"
         ]
      }
   }
}
```

`PublishResultsOnTopic`

(Facoltativo) L'argomento su cui si desidera pubblicare i risultati dell'inferenza. Se modificate questo valore, dovete modificare anche il valore del `accessControl` parametro `resources` in modo che corrisponda al nome dell'argomento personalizzato.

Impostazione predefinita: `ml/dlr/object-detection`

`Accelerator`

L'acceleratore che desideri utilizzare. I valori supportati sono `cpu` e `gpu`.

I modelli di esempio nel componente del modello dipendente supportano solo l'accelerazione della CPU. Per utilizzare l'accelerazione GPU con un modello personalizzato diverso, create un componente del modello personalizzato per sovrascrivere il componente del modello pubblico.

Impostazione predefinita: `cpu`

`ImageDirectory`

(Facoltativo) Il percorso della cartella sul dispositivo in cui i componenti di inferenza leggono le immagini. È possibile modificare questo valore in qualsiasi posizione del dispositivo a cui si dispone di accesso in lettura/scrittura.

Impostazione predefinita: */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*`/object_detection/sample_images/`

> **Note**
>
> Se impostate il valore di `UseCamera` to`true`, questo parametro di configurazione viene ignorato.

`ImageName`

(Facoltativo) Il nome dell'immagine che il componente di inferenza utilizza come input per effettuare una previsione. Il componente cerca l'immagine nella cartella specificata in. `ImageDirectory` Per impostazione predefinita, il componente utilizza l'immagine di esempio nella directory delle immagini predefinita. AWS IoT Greengrasssupporta i seguenti formati di immagine: `jpeg`jpg,png, enpy.

Impostazione predefinita: `objects.jpg`

> **Note**
>
> Se si imposta il valore di `UseCamera` to`true`, questo parametro di configurazione viene ignorato.

`InferenceInterval`

(Facoltativo) Il tempo in secondi tra ogni previsione effettuata dal codice di inferenza. Il codice di inferenza di esempio viene eseguito all'infinito e ripete le previsioni all'intervallo di tempo specificato. Ad esempio, è possibile impostare un intervallo più breve se si desidera utilizzare le immagini scattate da una fotocamera per la previsione in tempo reale.

Impostazione predefinita: `3600`

`ModelResourceKey`

(Facoltativo) I modelli utilizzati nel componente del modello pubblico dipendente. Modificate questo parametro solo se sostituite il componente del modello pubblico con un componente personalizzato.

Impostazione predefinita:

```
{
```

```
    "armv7l": "DLR-yolo3-armv7l-cpu-ObjectDetection",
    "aarch64": "DLR-yolo3-aarch64-gpu-ObjectDetection",
    "x86_64": "DLR-yolo3-x86_64-cpu-ObjectDetection",
    "windows": "DLR-resnet50-win-cpu-ObjectDetection"
}
```

`UseCamera`

(Facoltativo) Valore di stringa che definisce se utilizzare le immagini di una fotocamera collegata al dispositivo principale Greengrass. I valori supportati sono `true` e `false`.

Quando impostate questo valore su`true`, il codice di inferenza di esempio accede alla fotocamera del dispositivo ed esegue l'inferenza localmente sull'immagine acquisita. I valori dei `ImageDirectory` parametri `ImageName` and vengono ignorati. Assicuratevi che l'utente che esegue questo componente abbia accesso in lettura/scrittura alla posizione in cui la fotocamera memorizza le immagini acquisite.

Impostazione predefinita: `false`

> **Note**
>
> Quando si visualizza la ricetta di questo componente, il parametro di `UseCamera` configurazione non viene visualizzato nella configurazione predefinita. Tuttavia, è possibile modificare il valore di questo parametro in un aggiornamento di fusione della configurazione quando si distribuisce il componente.
> Se impostate su `UseCameratrue`, dovete anche creare un collegamento simbolico per consentire al componente di inferenza di accedere alla telecamera dall'ambiente virtuale creato dal componente runtime. Per ulteriori informazioni sull'utilizzo di una telecamera con i componenti di inferenza di esempio, vedere. Aggiornamento delle configurazioni dei componenti

2.0.x

`MLRootPath`

(Facoltativo) Il percorso della cartella sui dispositivi principali Linux in cui i componenti di inferenza leggono le immagini e scrivono i risultati dell'inferenza. È possibile modificare questo valore in qualsiasi posizione del dispositivo a cui l'utente che esegue questo componente ha accesso in lettura/scrittura.

Impostazione predefinita: */greengrass/v2*/work/variant.DLR/greengrass_ml

Impostazione predefinita: */greengrass/v2*/work/variant.TensorFlowLite/greengrass_ml

`Accelerator`

Non modificare. Attualmente, l'unico valore supportato per l'acceleratore è`cpu`, poiché i modelli nei componenti del modello dipendente vengono compilati solo per l'acceleratore CPU.

`ImageName`

(Facoltativo) Il nome dell'immagine che il componente di inferenza utilizza come input per una previsione di creazione. Il componente cerca l'immagine nella cartella specificata in. `ImageDirectory` La posizione predefinita è*MLRootPath*`/images`. AWS IoT Greengrasssupporta i seguenti formati di immagine: `jpeg`jpg,png, enpy.

Impostazione predefinita: `objects.jpg`

`InferenceInterval`

(Facoltativo) Il tempo in secondi tra ogni previsione effettuata dal codice di inferenza. Il codice di inferenza di esempio viene eseguito all'infinito e ripete le previsioni all'intervallo di tempo specificato. Ad esempio, è possibile impostare un intervallo più breve se si desidera utilizzare le immagini scattate da una fotocamera per la previsione in tempo reale.

Impostazione predefinita: `3600`

`ModelResourceKey`

(Facoltativo) I modelli utilizzati nel componente del modello pubblico dipendente. Modificate questo parametro solo se sostituite il componente del modello pubblico con un componente personalizzato.

Impostazione predefinita:

```
{
    armv7l: "DLR-yolo3-armv7l-cpu-ObjectDetection",
    x86_64: "DLR-yolo3-x86_64-cpu-ObjectDetection"
}
```

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.DLRObjectDetection.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.DLRObjectDetection.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.greengrass.DLRObjectDetection.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.DLRObjectDetection.log -Tail 10
   -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.1.13	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.1.12	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.11	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.10	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.9	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.

Versione	Modifiche
2.1.8	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.7	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.1.5	Componente rilasciato in tutto. Regioni AWS
2.1.4	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus. Questa versione non è disponibile in Europa (Londra) (). `eu-west-2`
2.1.3	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.1.2	Correzioni di bug e miglioramenti • Risolve un problema di ridimensionamento dell'immagine che causava riquadri di delimitazione imprecisi nei risultati di inferenza del rilevamen to di oggetti DLR di esempio.

Versione	Modifiche
2.1.1	Nuove funzionalità • Usa Deep Learning Runtime v1.6.0. • Aggiungi il supporto per il rilevamento di oggetti di esempio sulle piattaforme Armv8 (AArch64). Ciò estende il supporto dell'appr endimento automatico per i dispositivi core Greengrass che eseguono NVIDIA Jetson, come Jetson Nano. • Abilita l'integrazione della fotocamera per l'inferenza dei campioni. Utilizzate il nuovo parametro di `UseCamera` configurazione per consentire al codice di inferenza di esempio di accedere alla telecamera sul dispositivo principale Greengrass ed eseguire l'inferenza localmente sull'immagine acquisita. • Aggiungi il supporto per la pubblicazione dei risultati di inferenza a. Cloud AWS Utilizzate il nuovo parametro di `PublishResultsOnTo pic` configurazione per specificare l'argomento su cui desiderate pubblicare i risultati. • Aggiungete il nuovo parametro di `ImageDirectory` configurazione che consente di specificare una directory personalizzata per l'immagine su cui desiderate eseguire l'inferenza. Correzioni di bug e miglioramenti • Scrivi i risultati dell'inferenza nel file di registro del componente anziché in un file di inferenza separato. • Utilizzate il modulo di registrazione del software AWS IoT Greengrass Core per registrare l'output dei componenti. • Utilizzate il SDK per dispositivi AWS IoT per leggere la configurazione del componente e applicare le modifiche alla configurazione.
2.0.4	Versione iniziale.

Archivio modelli di classificazione delle immagini DLR

L'archivio dei modelli di classificazione delle immagini DLR è un componente del modello di apprendimento automatico che contiene ResNet -50 modelli pre-addestrati come artefatti

Greengrass. I modelli pre-addestrati utilizzati in questo componente vengono recuperati da GluonCV Model Zoo e compilati utilizzando Neo Deep Learning Runtime. SageMaker

Il componente di inferenza per la classificazione delle immagini DLR utilizza questo componente come dipendenza per l'origine del modello. Per utilizzare un modello DLR personalizzato, create una versione personalizzata di questo componente del modello e includete il modello personalizzato come elemento del componente. È possibile utilizzare la ricetta di questo componente come modello per creare componenti del modello personalizzati.

> **Note**
>
> Il nome del componente di archiviazione del modello di classificazione delle immagini DLR varia a seconda della versione. Il nome del componente per la versione 2.1.x e le versioni successive è. `variant.DLR.ImageClassification.ModelStore` Il nome del componente per la versione 2.0.x è. `variant.ImageClassification.ModelStore`

Argomenti

- Versioni
- Type
- Sistema operativo
- Requisiti
- Dipendenze
- Configurazione
- File di registro locale
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x () `variant.DLR.ImageClassification.ModelStore`
- 2,0x () `variant.ImageClassification.ModelStore`

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

Per abilitare lo stack di telecamere precedente

1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

   ```
   sudo raspi-config
   ```

2. Seleziona Opzioni di interfaccia.
3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
4. Riavvia il dispositivo Raspberry Pi.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.12

La tabella seguente elenca le dipendenze per la versione 2.1.12 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili

2.1.11

La tabella seguente elenca le dipendenze per la versione 2.1.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili

2.1.10

La tabella seguente elenca le dipendenze per la versione 2.1.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili

2.1.9

La tabella seguente elenca le dipendenze per la versione 2.1.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili

2.1.8

La tabella seguente elenca le dipendenze per la versione 2.1.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili

2.1.5

La tabella seguente elenca le dipendenze per la versione 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili

2.1.4

La tabella seguente elenca le dipendenze per la versione 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili

2.0.x

La tabella seguente elenca le dipendenze per la versione 2.0.x di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	~2.0.0	Flessibili

Configurazione

Questo componente non ha parametri di configurazione.

File di registro locale

Questo componente non emette registri.

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.1.12	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.1.11	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.10	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.9	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.8	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.

Versione	Modifiche
2.1.7	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.5	Nuove funzionalità • Aggiunge esempi di modelli di classificazione delle immagini per i dispositivi Windows principali. • Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.1.4	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.1.1	Nuove funzionalità • Aggiungi un modello di classificazione delle immagini ResNet -50 di esempio per le piattaforme Armv8 (AArch64). Ciò estende il supporto dell'apprendimento automatico per i dispositivi core Greengrass che eseguono NVIDIA Jetson, come Jetson Nano.
2.0.4	Versione iniziale.

Archivio modelli DLR per il rilevamento di oggetti

L'archivio modelli di rilevamento degli oggetti DLR è un componente del modello di apprendimento automatico che contiene modelli YOLov3 preaddestrati come artefatti Greengrass. I modelli di esempio utilizzati in questo componente vengono recuperati da GluonCV Model Zoo e compilati utilizzando Neo Deep Learning Runtime. SageMaker

Il componente di inferenza per il rilevamento degli oggetti DLR utilizza questo componente come dipendenza per l'origine del modello. Per utilizzare un modello DLR personalizzato, create una versione personalizzata di questo componente del modello e includete il modello personalizzato come elemento del componente. È possibile utilizzare la ricetta di questo componente come modello per creare componenti del modello personalizzati.

> **Note**
>
> Il nome del componente DLR Object Detection Model Store varia a seconda della versione. Il nome del componente per la versione 2.1.x e versioni successive è. `variant.DLR.ObjectDetection.ModelStore` Il nome del componente per la versione 2.0.x è. `variant.ObjectDetection.ModelStore`

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x
- 2,0x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

    Per abilitare lo stack di telecamere precedente

    1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleziona Opzioni di interfaccia.
    3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
    4. Riavvia il dispositivo Raspberry Pi.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.13

La tabella seguente elenca le dipendenze per la versione 2.1.13 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili

2.1.12

La tabella seguente elenca le dipendenze per la versione 2.1.12 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili

2.1.11

La tabella seguente elenca le dipendenze per la versione 2.1.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili

2.1.10

La tabella seguente elenca le dipendenze per la versione 2.1.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili

2.1.9

La tabella seguente elenca le dipendenze per la versione 2.1.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili

2.1.8

La tabella seguente elenca le dipendenze per la versione 2.1.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili

2.1.5 and 2.1.6

La tabella seguente elenca le dipendenze per le versioni 2.1.5 e 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili

2.1.4

La tabella seguente elenca le dipendenze per la versione 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili

2.0.x

La tabella seguente elenca le dipendenze per la versione 2.0.x di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	~2.0.0	Flessibili

Configurazione

Questo componente non ha parametri di configurazione.

File di registro locale

Questo componente non emette registri.

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.1.13	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.1.12	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.11	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.10	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.9	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.8	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.7	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.6	Aggiunge un modello di CPU per risolvere un problema sui dispositivi Armv8 (AArch64).
2.1.5	Nuove funzionalità • Aggiunge modelli di rilevamento di oggetti di esempio per i dispositivi Windows core.

Versione	Modifiche
	Correzioni di bug e miglioramenti • Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.1.4	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.1.1	Nuove funzionalità • Aggiungi un modello di rilevamento di oggetti YOLov3 di esempio per piattaforme Armv8 (AArch64). Ciò estende il supporto dell'appr endimento automatico per i dispositivi core Greengrass che eseguono NVIDIA Jetson, come Jetson Nano.
2.0.4	Versione iniziale.

Runtime DLR

Il componente runtime DLR (`variant.DLR`) contiene uno script che installa Deep Learning Runtime (DLR) e le relative dipendenze in un ambiente virtuale sul dispositivo. I Rilevamento di oggetti DLR componenti Classificazione delle immagini DLR and utilizzano questo componente come dipendenza per l'installazione di DLR. La versione del componente 1.6.x installa DLR v1.6.0 e la versione del componente 1.3.x installa DLR v1.3.0.

Per utilizzare un runtime diverso, è possibile utilizzare la ricetta di questo componente come modello per creare un componente di machine learning personalizzato.

Argomenti

- Versioni
- Type
- Sistema operativo
- Requisiti
- Dipendenze
- Configurazione

- Utilizzo
- File di registro locale
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 1.6.x
- 1.3.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

```
sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
 libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

    Per abilitare lo stack di telecamere precedente

    1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleziona Opzioni di interfaccia.
    3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
    4. Riavvia il dispositivo Raspberry Pi.

Endpoint e porte

Per impostazione predefinita, questo componente utilizza uno script di installazione per installare i pacchetti utilizzando i `pip` comandi `apt`,`yum`,`brew`, e, a seconda della piattaforma utilizzata dal dispositivo principale. Questo componente deve essere in grado di eseguire le richieste in uscita verso vari indici e repository di pacchetti per eseguire lo script di installazione. Per consentire il traffico in uscita di questo componente attraverso un proxy o un firewall, è necessario identificare gli endpoint degli indici e degli archivi dei pacchetti a cui il dispositivo principale si connette per l'installazione.

Quando identificate gli endpoint necessari per lo script di installazione di questo componente, tenete presente quanto segue:

- Gli endpoint dipendono dalla piattaforma del dispositivo principale. Ad esempio, un dispositivo principale che esegue Ubuntu utilizza `apt` anziché `yum` o`brew`. Inoltre, i dispositivi che utilizzano lo

stesso indice di pacchetti potrebbero avere elenchi di sorgenti diversi, quindi potrebbero recuperare pacchetti da repository diversi.

- Gli endpoint potrebbero differire tra più dispositivi che utilizzano lo stesso indice di pacchetti, poiché ogni dispositivo ha i propri elenchi di sorgenti che definiscono dove recuperare i pacchetti.
- Gli endpoint potrebbero cambiare nel tempo. Ogni indice dei pacchetti fornisce gli URL dei repository in cui si scaricano i pacchetti e il proprietario di un pacchetto può modificare gli URL forniti dall'indice dei pacchetti.

Per ulteriori informazioni sulle dipendenze installate da questo componente e su come disabilitare lo script di installazione, vedete il parametro di configurazione. UseInstaller

Per ulteriori informazioni sugli endpoint e le porte necessari per il funzionamento di base, vedere. Consenti il traffico dei dispositivi tramite un proxy o un firewall

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

1.6.11 and 1.6.12

La tabella seguente elenca le dipendenze per le versioni 1.6.11 e 1.6.12 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <3.0.0	Flessibili

1.6.10

La tabella seguente elenca le dipendenze per la versione 1.6.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili

1.6.9

La tabella seguente elenca le dipendenze per la versione 1.6.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili

1.6.8

La tabella seguente elenca le dipendenze per la versione 1.6.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili

1.6.6 and 1.6.7

La tabella seguente elenca le dipendenze per le versioni 1.6.6 e 1.6.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili

1.6.4 and 1.6.5

La tabella seguente elenca le dipendenze per le versioni 1.6.4 e 1.6.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili

1.6.3

La tabella seguente elenca le dipendenze per la versione 1.6.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili

1.6.2

La tabella seguente elenca le dipendenze per la versione 1.6.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

1.6.1

La tabella seguente elenca le dipendenze per la versione 1.6.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili

1.3.x

La tabella seguente elenca le dipendenze per la versione 1.3.x di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	~2.0.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`MLRootPath`

(Facoltativo) Il percorso della cartella sui dispositivi principali di Linux in cui i componenti di inferenza leggono le immagini e scrivono i risultati dell'inferenza. È possibile modificare questo valore in qualsiasi posizione del dispositivo a cui l'utente che esegue questo componente ha accesso in lettura/scrittura.

Impostazione predefinita: */greengrass/v2*`/work/variant.DLR/greengrass_ml`

`WindowsMLRootPath`

Questa funzionalità è disponibile nella versione 1.6.6 e successive di questo componente.

(Facoltativo) Il percorso della cartella sul dispositivo principale di Windows in cui i componenti di inferenza leggono le immagini e scrivono i risultati dell'inferenza. È possibile modificare questo valore in qualsiasi posizione del dispositivo a cui l'utente che esegue questo componente ha accesso in lettura/scrittura.

Impostazione predefinita: *C:\greengrass\v2*`\\work\\variant.DLR\\greengrass_ml`

`UseInstaller`

(Facoltativo) Valore di stringa che definisce se utilizzare lo script di installazione in questo componente per installare DLR e le relative dipendenze. I valori supportati sono `true` e `false`.

Imposta questo valore su `false` se desideri utilizzare uno script personalizzato per l'installazione DLR o se desideri includere dipendenze di runtime in un'immagine Linux predefinita. Per utilizzare questo componente con i componenti di inferenza DLR AWS forniti, installate le seguenti librerie, comprese le eventuali dipendenze, e mettetele a disposizione dell'utente del sistema, ad esempio chi esegue i `ggc_user` componenti ML.

- Python 3.7 o successivo, incluso `pip` per la tua versione di Python.
- Deep Learning Runtime v1.6.0
- NumPy.
- OpenCV-Python.
- SDK per dispositivi AWS IoTv2 per Python.
- AWSPython Common Runtime (CRT).

- Picamera (solo per dispositivi Raspberry Pi).
- `awscammodulo` (per AWS DeepLens dispositivi).
- LibGL (per dispositivi Linux)

Impostazione predefinita: `true`

Utilizzo

Utilizzate questo componente con il parametro `UseInstaller` di configurazione impostato per `true` installare DLR e le sue dipendenze sul dispositivo. Il componente configura un ambiente virtuale sul dispositivo che include OpenCV NumPy e le librerie necessarie per DLR.

> **Note**
>
> Lo script di installazione di questo componente installa anche le versioni più recenti di librerie di sistema aggiuntive necessarie per configurare l'ambiente virtuale sul dispositivo e per utilizzare il framework di machine learning installato. Ciò potrebbe aggiornare le librerie di sistema esistenti sul dispositivo. Consulta la tabella seguente per l'elenco delle librerie installate da questo componente per ogni sistema operativo supportato. Se desiderate personalizzare questo processo di installazione, impostate il parametro di `UseInstaller` configurazione su e sviluppate il vostro script di installazione. `false`

Piattaforma	Librerie installate sul sistema del dispositivo	Librerie installate nell'ambiente virtuale
Armv7l	`build-essential` , `cmake`, `ca-certificates` , `git`	`setuptools` , `wheel`
Amazon Linux 2	`mesa-libGL`	Nessuno
Ubuntu	`wget`	Nessuno

Quando distribuite il componente di inferenza, questo componente di runtime verifica innanzitutto se sul dispositivo sono già installati DLR e le relative dipendenze e, in caso contrario, li installa automaticamente.

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/variant.DLR.log
```

Windows

```
C:\greengrass\v2\logs\variant.DLR.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/variant.DLR.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\variant.DLR.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
1.6.12	Correzioni di bug e miglioramenti • Corregge lo script di installazione per gli utenti del sistema operativo Windows.
1.6.11	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.

Versione	Modifiche
1.6.10	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
1.6.9	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
1.6.8	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
1.6.7	Correzioni di bug e miglioramenti • Aggiorna lo script di `UseInstaller` installazione per installare LibGL, che non è disponibile per impostazione predefinita su alcune piattaforme Linux. • Aggiorna lo script di `UseInstaller` installazione per utilizzare sempre Python 3.9 nell'ambiente virtuale di questo componente. Questa modifica aiuta a garantire la compatibilità con altre librerie.
1.6.6	Nuove funzionalità • Aggiunge il supporto per i dispositivi principali che eseguono Windows. • Aggiunge il nuovo parametro di `WindowsMLRootPath` configura zione che è possibile utilizzare per configurare la cartella dei risultati di inferenza sui dispositivi principali di Windows.
1.6.5	Nuove funzionalità • Aggiunge il nuovo parametro di `UseInstaller` configurazione che è possibile utilizzare per disabilitare lo script di installazione in questo componente.
1.6.4	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
1.6.3	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
1.6.2	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.

Versione	Modifiche
1.6.1	Nuove funzionalità • Installa Deep Learning Runtime v1.6.0 e le sue dipendenze. • Aggiungi il supporto per l'installazione di DLR su piattaforme Armv8 (AArch64). Ciò estende il supporto dell'apprendimento automatico per i dispositivi core Greengrass che eseguono NVIDIA Jetson, come Jetson Nano. Correzioni di bug e miglioramenti • Installa il SDK per dispositivi AWS IoT nell'ambiente virtuale per leggere la configurazione del componente e applicare le modifiche alla configura zione. • Correzioni e miglioramenti aggiuntivi di bug minori.
1.3.2	Versione iniziale. Installa DLR v1.3.0.

TensorFlow Classificazione delle immagini Lite

Il componente di classificazione delle immagini TensorFlow Lite (`aws.greengrass.TensorFlowLiteImageClassification`) contiene un codice di inferenza di esempio per eseguire l'inferenza della classificazione delle immagini utilizzando il runtime TensorFlow Lite e un modello quantizzato MobileNet 1.0 pre-addestrato di esempio. Questo componente utilizza la variante TensorFlow Archivio di modelli di classificazione delle immagini Lite e TensorFlow Runtime Lite i componenti come dipendenze per scaricare il runtime TensorFlow Lite e il modello di esempio.

Per utilizzare questo componente di inferenza con un modello TensorFlow Lite personalizzato, create una versione personalizzata del componente dipendente del Model Store. Per utilizzare il proprio codice di inferenza personalizzato, è possibile utilizzare la ricetta di questo componente come modello per creare un componente di inferenza personalizzato.

Argomenti

- Versioni
- Type
- Sistema operativo

- Requisiti
- Dipendenze
- Configurazione
- File di registro locale
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

```
sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
 libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

    Per abilitare lo stack di telecamere precedente

    1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleziona Opzioni di interfaccia.
    3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
    4. Riavvia il dispositivo Raspberry Pi.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.11

La tabella seguente elenca le dipendenze per la versione 2.1.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.10

La tabella seguente elenca le dipendenze per la versione 2.1.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.9

La tabella seguente elenca le dipendenze per la versione 2.1.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.8

La tabella seguente elenca le dipendenze per la versione 2.1.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili

Dipendenza	Versioni compatibili	Tipo di dipendenza
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.5

La tabella seguente elenca le dipendenze per la versione 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.4

La tabella seguente elenca le dipendenze per la versione 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

Dipendenza	Versioni compatibili	Tipo di dipendenza
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`accessControl`

(Facoltativo) L'oggetto che contiene la politica di autorizzazione che consente al componente di pubblicare messaggi nell'argomento delle notifiche predefinito.

Impostazione predefinita:

```
{
   "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.TensorFlowLiteImageClassification:mqttproxy:1": {
         "policyDescription": "Allows access to publish via topic ml/tflite/image-
classification.",
```

```
            "operations": [
               "aws.greengrass#PublishToIoTCore"
            ],
            "resources": [
               "ml/tflite/image-classification"
            ]
         }
      }
   }
```

`PublishResultsOnTopic`

(Facoltativo) L'argomento su cui si desidera pubblicare i risultati dell'inferenza. Se modificate questo valore, dovete modificare anche il valore del `accessControl` parametro `resources` in modo che corrisponda al nome dell'argomento personalizzato.

Impostazione predefinita: `ml/tflite/image-classification`

`Accelerator`

L'acceleratore che desideri utilizzare. I valori supportati sono `cpu` e `gpu`.

I modelli di esempio nel componente del modello dipendente supportano solo l'accelerazione della CPU. Per utilizzare l'accelerazione GPU con un modello personalizzato diverso, create un componente del modello personalizzato per sovrascrivere il componente del modello pubblico.

Impostazione predefinita: `cpu`

`ImageDirectory`

(Facoltativo) Il percorso della cartella sul dispositivo in cui i componenti di inferenza leggono le immagini. È possibile modificare questo valore in qualsiasi posizione del dispositivo a cui si dispone di accesso in lettura/scrittura.

Impostazione predefinita: */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*`/image_classification/sample_images/`

> **Note**
>
> Se impostate il valore di `UseCamera` to`true`, questo parametro di configurazione viene ignorato.

`ImageName`

(Facoltativo) Il nome dell'immagine che il componente di inferenza utilizza come input per effettuare una previsione. Il componente cerca l'immagine nella cartella specificata in. `ImageDirectory` Per impostazione predefinita, il componente utilizza l'immagine di esempio nella directory delle immagini predefinita. AWS IoT Greengrasssupporta i seguenti formati di immagine: `jpegjpg`,`png`, `enpy`.

Impostazione predefinita: `cat.jpeg`

> **Note**
>
> Se si imposta il valore di `UseCamera` to`true`, questo parametro di configurazione viene ignorato.

`InferenceInterval`

(Facoltativo) Il tempo in secondi che intercorre tra ogni previsione effettuata dal codice di inferenza. Il codice di inferenza di esempio viene eseguito all'infinito e ripete le previsioni all'intervallo di tempo specificato. Ad esempio, è possibile impostare un intervallo più breve se si desidera utilizzare le immagini scattate da una fotocamera per la previsione in tempo reale.

Impostazione predefinita: `3600`

`ModelResourceKey`

(Facoltativo) I modelli utilizzati nel componente del modello pubblico dipendente. Modificate questo parametro solo se sostituite il componente del modello pubblico con un componente personalizzato.

Impostazione predefinita:

```
{
    "model": "TensorFlowLite-Mobilenet"
}
```

`UseCamera`

(Facoltativo) Valore di stringa che definisce se utilizzare le immagini di una fotocamera collegata al dispositivo principale Greengrass. I valori supportati sono `true` e `false`.

Quando impostate questo valore su`true`, il codice di inferenza di esempio accede alla fotocamera del dispositivo ed esegue l'inferenza localmente sull'immagine acquisita. I valori dei `ImageDirectory` parametri `ImageName` and vengono ignorati. Assicuratevi che l'utente che esegue questo componente abbia accesso in lettura/scrittura alla posizione in cui la fotocamera memorizza le immagini acquisite.

Impostazione predefinita: `false`

> **Note**
>
> Quando si visualizza la ricetta di questo componente, il parametro di `UseCamera` configurazione non viene visualizzato nella configurazione predefinita. Tuttavia, è possibile modificare il valore di questo parametro in un aggiornamento di fusione della configurazione quando si distribuisce il componente.
> Se impostate su `UseCameratrue`, dovete anche creare un collegamento simbolico per consentire al componente di inferenza di accedere alla telecamera dall'ambiente virtuale creato dal componente runtime. Per ulteriori informazioni sull'utilizzo di una telecamera con i componenti di inferenza di esempio, vedere. Aggiornamento delle configurazioni dei componenti

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.TensorFlowLiteImageClassification.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.TensorFlowLiteImageClassification.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

Linux

```
sudo tail -f /greengrass/v2/logs/
aws.greengrass.TensorFlowLiteImageClassification.log
```

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs
\aws.greengrass.TensorFlowLiteImageClassification.log -Tail 10 -Wait
```

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
2.1.11	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.1.10	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.9	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.8	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.7	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.5	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.4	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.1.1	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.1.0	Versione iniziale.

TensorFlow Rilevamento di oggetti Lite

Il componente TensorFlow Lite object detection (`aws.greengrass.TensorFlowLiteObjectDetection`) contiene un codice di inferenza di esempio per eseguire l'inferenza del rilevamento di oggetti utilizzando TensorFlow Lite e un modello Single Shot Detection (SSD) 1.0 di esempio pre-addestrato. MobileNet Questo componente utilizza la variante TensorFlow Archivio modelli Lite per il rilevamento di oggetti e i TensorFlow Runtime Lite componenti come dipendenze per scaricare TensorFlow Lite e il modello di esempio.

Per utilizzare questo componente di inferenza con un modello TensorFlow Lite personalizzato, puoi creare una versione personalizzata del componente dipendente del Model Store. Per utilizzare il tuo codice di inferenza personalizzato, usa la ricetta di questo componente come modello per creare un componente di inferenza personalizzato.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

    Per abilitare lo stack di telecamere precedente

    1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleziona Opzioni di interfaccia.

3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
4. Riavvia il dispositivo Raspberry Pi.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.11

La tabella seguente elenca le dipendenze per la versione 2.1.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.10

La tabella seguente elenca le dipendenze per la versione 2.1.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi

Dipendenza	Versioni compatibili	Tipo di dipendenza
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.9

La tabella seguente elenca le dipendenze per la versione 2.1.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.8

La tabella seguente elenca le dipendenze per la versione 2.1.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.5

La tabella seguente elenca le dipendenze per la versione 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.4

La tabella seguente elenca le dipendenze per la versione 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili

Dipendenza	Versioni compatibili	Tipo di dipendenza
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili
TensorFlow Archivio di modelli di classificazione delle immagini Lite	>=2.1.0 <2.2.0	Rigidi
TensorFlow Leggero	>=2,5,0 <2,6,0	Rigidi

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`accessControl`

(Facoltativo) L'oggetto che contiene la politica di autorizzazione che consente al componente di pubblicare messaggi nell'argomento delle notifiche predefinito.

Impostazione predefinita:

```
{
   "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.TensorFlowLiteObjectDetection:mqttproxy:1": {
         "policyDescription": "Allows access to publish via topic ml/tflite/object-
detection.",
         "operations": [
            "aws.greengrass#PublishToIoTCore"
         ],
         "resources": [
            "ml/tflite/object-detection"
         ]
      }
   }
}
```

`PublishResultsOnTopic`

(Facoltativo) L'argomento su cui si desidera pubblicare i risultati dell'inferenza. Se modificate questo valore, dovete modificare anche il valore del `accessControl` parametro `resources` in modo che corrisponda al nome dell'argomento personalizzato.

Impostazione predefinita: `ml/tflite/object-detection`

`Accelerator`

L'acceleratore che desideri utilizzare. I valori supportati sono `cpu` e `gpu`.

I modelli di esempio nel componente del modello dipendente supportano solo l'accelerazione della CPU. Per utilizzare l'accelerazione GPU con un modello personalizzato diverso, create un componente del modello personalizzato per sovrascrivere il componente del modello pubblico.

Impostazione predefinita: `cpu`

`ImageDirectory`

(Facoltativo) Il percorso della cartella sul dispositivo in cui i componenti di inferenza leggono le immagini. È possibile modificare questo valore in qualsiasi posizione del dispositivo a cui si dispone di accesso in lettura/scrittura.

Impostazione predefinita: */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*`/object_detection/sample_images/`

> **Note**
>
> Se impostate il valore di `UseCamera` to`true`, questo parametro di configurazione viene ignorato.

`ImageName`

(Facoltativo) Il nome dell'immagine che il componente di inferenza utilizza come input per effettuare una previsione. Il componente cerca l'immagine nella cartella specificata in. `ImageDirectory` Per impostazione predefinita, il componente utilizza l'immagine di esempio nella directory delle immagini predefinita. AWS IoT Greengrasssupporta i seguenti formati di immagine: `jpeg`jpg,png, `enpy`.

Impostazione predefinita: `objects.jpg`

> **Note**
>
> Se si imposta il valore di `UseCamera` to`true`, questo parametro di configurazione viene ignorato.

`InferenceInterval`

(Facoltativo) Il tempo in secondi tra ogni previsione effettuata dal codice di inferenza. Il codice di inferenza di esempio viene eseguito all'infinito e ripete le previsioni all'intervallo di tempo specificato. Ad esempio, è possibile impostare un intervallo più breve se si desidera utilizzare le immagini scattate da una fotocamera per la previsione in tempo reale.

Impostazione predefinita: `3600`

`ModelResourceKey`

(Facoltativo) I modelli utilizzati nel componente del modello pubblico dipendente. Modificate questo parametro solo se sostituite il componente del modello pubblico con un componente personalizzato.

Impostazione predefinita:

```
{
    "model": "TensorFlowLite-SSD"
}
```

`UseCamera`

(Facoltativo) Valore di stringa che definisce se utilizzare le immagini di una fotocamera collegata al dispositivo principale Greengrass. I valori supportati sono `true` e `false`.

Quando impostate questo valore su`true`, il codice di inferenza di esempio accede alla fotocamera del dispositivo ed esegue l'inferenza localmente sull'immagine acquisita. I valori dei `ImageDirectory` parametri `ImageName` and vengono ignorati. Assicuratevi che l'utente che esegue questo componente abbia accesso in lettura/scrittura alla posizione in cui la fotocamera memorizza le immagini acquisite.

Impostazione predefinita: `false`

> **Note**
>
> Quando si visualizza la ricetta di questo componente, il parametro di `UseCamera` configurazione non viene visualizzato nella configurazione predefinita. Tuttavia, è possibile modificare il valore di questo parametro in un aggiornamento di fusione della configurazione quando si distribuisce il componente.
> Se impostate su `UseCameratrue`, dovete anche creare un collegamento simbolico per consentire al componente di inferenza di accedere alla telecamera dall'ambiente virtuale creato dal componente runtime. Per ulteriori informazioni sull'utilizzo di una telecamera con i componenti di inferenza di esempio, vedere. Aggiornamento delle configurazioni dei componenti

> **Note**
>
> Quando si visualizza la ricetta di questo componente, il parametro di `UseCamera` configurazione non viene visualizzato nella configurazione predefinita. Tuttavia, è possibile modificare il valore di questo parametro in un aggiornamento di fusione della configurazione quando si distribuisce il componente.
> Se impostate su `UseCameratrue`, dovete anche creare un collegamento simbolico per consentire al componente di inferenza di accedere alla telecamera dall'ambiente virtuale creato dal componente runtime. Per ulteriori informazioni sull'utilizzo di una telecamera con i componenti di inferenza di esempio, vedere. Aggiornamento delle configurazioni dei componenti

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.TensorFlowLiteObjectDetection.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.TensorFlowLiteObjectDetection.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/
  aws.greengrass.TensorFlowLiteObjectDetection.log
  ```

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs
\aws.greengrass.TensorFlowLiteObjectDetection.log -Tail 10 -Wait
```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.1.11	Versione aggiornata per Greengrass nucleus versione 2.12.0.
2.1.10	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.9	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.8	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.7	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.5	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.4	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.1.1	Correzioni di bug e miglioramenti • Risolve un problema di ridimensionamento dell'immagine che causava riquadri di delimitazione imprecisi nei risultati di inferenza del rilevamento di oggetti TensorFlow Lite di esempio.
2.1.0	Versione iniziale.

TensorFlow Archivio di modelli di classificazione delle immagini Lite

Il modello di classificazione delle immagini TensorFlow Lite (`variant.TensorFlowLite.ImageClassification.ModelStore`) è un componente del modello di apprendimento automatico che contiene un modello MobileNet v1 pre-addestrato come artefatto Greengrass. Il modello di esempio utilizzato in questo componente viene recuperato dall'Hub e implementato utilizzando Lite. TensorFlow TensorFlow

Il componente di TensorFlow Classificazione delle immagini Lite inferenza utilizza questo componente come dipendenza per l'origine del modello. Per utilizzare un modello TensorFlow Lite personalizzato, create una versione personalizzata di questo componente del modello e includete il modello personalizzato come elemento del componente. È possibile utilizzare la ricetta di questo componente come modello per creare componenti del modello personalizzati.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

    Per abilitare lo stack di telecamere precedente

    1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleziona Opzioni di interfaccia.

3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
4. Riavvia il dispositivo Raspberry Pi.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.11

La tabella seguente elenca le dipendenze per la versione 2.1.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili

2.1.10

La tabella seguente elenca le dipendenze per la versione 2.1.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili

2.1.9

La tabella seguente elenca le dipendenze per la versione 2.1.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili

2.1.8

La tabella seguente elenca le dipendenze per la versione 2.1.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili

2.1.5

La tabella seguente elenca le dipendenze per la versione 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili

2.1.4

La tabella seguente elenca le dipendenze per la versione 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili

Configurazione

Questo componente non ha parametri di configurazione.

File di registro locale

Questo componente non emette registri.

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.1.11	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.1.10	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.9	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.8	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.7	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.5	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.4	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.1.1	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.1.0	Versione iniziale.

TensorFlow Archivio modelli Lite per il rilevamento di oggetti

Il TensorFlow Lite object detection model store (`variant.TensorFlowLite.ObjectDetection.ModelStore`) è un componente del modello

di machine learning che contiene un modello Single Shot Detection (SSD) pre-addestrato come MobileNet artefatto Greengrass. Il modello di esempio utilizzato in questo componente viene recuperato dall'Hub e implementato utilizzando Lite. TensorFlow TensorFlow

Il componente di inferenza per il rilevamento degli oggetti TensorFlow Lite utilizza questo componente come dipendenza per l'origine del modello. Per utilizzare un modello TensorFlow Lite personalizzato, create una versione personalizzata di questo componente del modello e includete il modello personalizzato come elemento del componente. È possibile utilizzare la ricetta di questo componente come modello per creare componenti del modello personalizzati.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

    Per abilitare lo stack di telecamere precedente

    1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleziona Opzioni di interfaccia.
    3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
    4. Riavvia il dispositivo Raspberry Pi.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.11

La tabella seguente elenca le dipendenze per la versione 2.1.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili

2.1.10

La tabella seguente elenca le dipendenze per la versione 2.1.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili

2.1.9

La tabella seguente elenca le dipendenze per la versione 2.1.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili

2.1.8

La tabella seguente elenca le dipendenze per la versione 2.1.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili

2.1.5

La tabella seguente elenca le dipendenze per la versione 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili

2.1.4

La tabella seguente elenca le dipendenze per la versione 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili

Configurazione

Questo componente non ha parametri di configurazione.

File di registro locale

Questo componente non emette registri.

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.1.11	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.1.10	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.9	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.8	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.7	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.5	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.4	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.1.1	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.1.0	Versione iniziale.

TensorFlow Runtime Lite

Il componente runtime TensorFlow Lite (`variant.TensorFlowLite`) contiene uno script che installa la versione TensorFlow Lite 2.5.0 e le relative dipendenze in un ambiente virtuale sul dispositivo. Il componente TensorFlow Lite per la classificazione delle immagini e il rilevamento degli oggetti TensorFlow Lite utilizzano questo componente di runtime come dipendenza per l'installazione di Lite. TensorFlow

> **Note**
>
> TensorFlow Il componente di runtime Lite v2.5.6 e versioni successive reinstalla le installazioni esistenti del runtime Lite e delle TensorFlow sue dipendenze. Questa reinstallazione aiuta a garantire che il dispositivo principale esegua versioni compatibili di Lite e delle sue dipendenze. TensorFlow

Per utilizzare un runtime diverso, puoi utilizzare la ricetta di questo componente come modello per creare un componente di apprendimento automatico personalizzato.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.5.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

  ```
  sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
   libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
  ```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

    Per abilitare lo stack di telecamere precedente

    1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleziona Opzioni di interfaccia.

3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
4. Riavvia il dispositivo Raspberry Pi.

Endpoint e porte

Per impostazione predefinita, questo componente utilizza uno script di installazione per installare i pacchetti utilizzando i `pip` comandi `apt`,`yum`,`brew`, e, a seconda della piattaforma utilizzata dal dispositivo principale. Questo componente deve essere in grado di eseguire le richieste in uscita verso vari indici e repository di pacchetti per eseguire lo script di installazione. Per consentire il traffico in uscita di questo componente attraverso un proxy o un firewall, è necessario identificare gli endpoint degli indici e degli archivi dei pacchetti a cui il dispositivo principale si connette per l'installazione.

Quando identificate gli endpoint necessari per lo script di installazione di questo componente, tenete presente quanto segue:

- Gli endpoint dipendono dalla piattaforma del dispositivo principale. Ad esempio, un dispositivo principale che esegue Ubuntu utilizza `apt` anziché `yum` o`brew`. Inoltre, i dispositivi che utilizzano lo stesso indice di pacchetti potrebbero avere elenchi di sorgenti diversi, quindi potrebbero recuperare pacchetti da repository diversi.
- Gli endpoint potrebbero differire tra più dispositivi che utilizzano lo stesso indice di pacchetti, poiché ogni dispositivo ha i propri elenchi di sorgenti che definiscono dove recuperare i pacchetti.
- Gli endpoint potrebbero cambiare nel tempo. Ogni indice dei pacchetti fornisce gli URL dei repository in cui si scaricano i pacchetti e il proprietario di un pacchetto può modificare gli URL forniti dall'indice dei pacchetti.

Per ulteriori informazioni sulle dipendenze installate da questo componente e su come disabilitare lo script di installazione, vedete il parametro di configurazione. UseInstaller

Per ulteriori informazioni sugli endpoint e le porte necessari per il funzionamento di base, vedere. Consenti il traffico dei dispositivi tramite un proxy o un firewall

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le

dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.5.14

La tabella seguente elenca le dipendenze per la versione 2.5.14 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili

2.5.13

La tabella seguente elenca le dipendenze per la versione 2.5.13 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili

2.5.12

La tabella seguente elenca le dipendenze per la versione 2.5.12 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili

2.5.11

La tabella seguente elenca le dipendenze per la versione 2.5.11 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili

2.5.10

La tabella seguente elenca le dipendenze per la versione 2.5.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili

2.5.9

La tabella seguente elenca le dipendenze per la versione 2.5.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili

2.5.8

La tabella seguente elenca le dipendenze per la versione 2.5.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili

2.5.5 - 2.5.7

La tabella seguente elenca le dipendenze per le versioni da 2.5.5 a 2.5.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili

2.5.3 and 2.5.4

La tabella seguente elenca le dipendenze per le versioni 2.5.3 e 2.5.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili

2.5.2

La tabella seguente elenca le dipendenze per la versione 2.5.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili

2.5.1

La tabella seguente elenca le dipendenze per la versione 2.5.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

2.5.0

La tabella seguente elenca le dipendenze per la versione 2.5.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`MLRootPath`

(Facoltativo) Il percorso della cartella sui dispositivi principali di Linux in cui i componenti di inferenza leggono le immagini e scrivono i risultati dell'inferenza. È possibile modificare questo valore in qualsiasi posizione del dispositivo a cui l'utente che esegue questo componente ha accesso in lettura/scrittura.

Impostazione predefinita: */greengrass/v2*`/work/variant.TensorFlowLite/greengrass_ml`

`WindowsMLRootPath`

Questa funzionalità è disponibile nella versione 1.6.6 e successive di questo componente.

(Facoltativo) Il percorso della cartella sul dispositivo principale di Windows in cui i componenti di inferenza leggono le immagini e scrivono i risultati dell'inferenza. È possibile modificare questo valore in qualsiasi posizione del dispositivo a cui l'utente che esegue questo componente ha accesso in lettura/scrittura.

Impostazione predefinita: *C:\greengrass\v2*`\\work\\variant.DLR\\greengrass_ml`

`UseInstaller`

(Facoltativo) Valore di stringa che definisce se utilizzare lo script di installazione in questo componente per installare TensorFlow Lite e le sue dipendenze. I valori supportati sono `true` e `false`.

Imposta questo valore su `false` se desideri utilizzare uno script personalizzato per l'installazione di TensorFlow Lite o se desideri includere dipendenze di runtime in un'immagine Linux predefinita. Per utilizzare questo componente con i componenti di inferenza TensorFlow Lite AWS forniti, installa le seguenti librerie, comprese le eventuali dipendenze, e rendile disponibili all'utente del sistema, ad esempio chi esegue i componenti `ggc_user` ML.

- Python 3.8 o successivo, incluso `pip` per la tua versione di Python
- TensorFlow Lite v2.5.0
- NumPy
- OpenCV-Python
- SDK per dispositivi AWS IoTv2 per Python
- AWSPython Common Runtime (CRT)

- Picamera (per dispositivi Raspberry Pi)
- awscammodulo (per AWS DeepLens dispositivi)
- LibGL (per dispositivi Linux)

Impostazione predefinita: `true`

Utilizzo

Utilizzate questo componente con il parametro `UseInstaller` di configurazione impostato per `true` installare TensorFlow Lite e le sue dipendenze sul dispositivo. Il componente configura un ambiente virtuale sul dispositivo che include OpenCV NumPy e le librerie necessarie per Lite. TensorFlow

> **Note**
>
> Lo script di installazione di questo componente installa anche le versioni più recenti di librerie di sistema aggiuntive necessarie per configurare l'ambiente virtuale sul dispositivo e per utilizzare il framework di machine learning installato. Ciò potrebbe aggiornare le librerie di sistema esistenti sul dispositivo. Consulta la tabella seguente per l'elenco delle librerie installate da questo componente per ogni sistema operativo supportato. Se desiderate personalizzare questo processo di installazione, impostate il parametro di `UseInstaller` configurazione su e sviluppate il vostro script di installazione. `false`

Piattaforma	Librerie installate sul sistema del dispositivo	Librerie installate nell'ambiente virtuale
Armv7l	`build-essential` , `cmake`, `ca-certificates` , `git`	`setuptools` , `wheel`
Amazon Linux 2	`mesa-libGL`	Nessuno
Ubuntu	`wget`	Nessuno

Quando si distribuisce il componente di inferenza, questo componente di runtime verifica innanzitutto se sul dispositivo sono già installati TensorFlow Lite e le relative dipendenze. In caso contrario, il componente di runtime li installa automaticamente.

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/variant.TensorFlowLite.log
```

Windows

```
C:\greengrass\v2\logs\variant.TensorFlowLite.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/variant.TensorFlowLite.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\variant.TensorFlowLite.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.5.14	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.5.13	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.5.12	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.

Versione	Modifiche
2.5.11	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.5.10	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.5.9	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.5.8	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.5.7	Correzioni di bug e miglioramenti • Aggiorna lo script di `UseInstaller` installazione per installare LibGL, che non è disponibile per impostazione predefinita su alcune piattaforme Linux. • Aggiorna lo script di `UseInstaller` installazione per utilizzare sempre Python 3.9 nell'ambiente virtuale di questo componente. Questa modifica aiuta a garantire la compatibilità con altre librerie.
2.5.6	Correzioni di bug e miglioramenti • Aggiorna questo componente per installare l'ultima patch di TensorFlow Lite 2.5.0 (`tflite-runtime-2.5.0.post1`), quindi puoi usare questo componente con Python 3.9. Se questo componente non riesce a installare quella patch, si installa al suo posto. `tflite-runtime-2.5.0` • Aggiorna questo componente per reinstallare le installazioni esistenti di TensorFlow Lite e le relative dipendenze. Questa modifica aiuta a garantire che il dispositivo principale esegua versioni compatibili di TensorFlow Lite e delle sue dipendenze.
2.5.5	Nuove funzionalità • Aggiunge il supporto per i dispositivi principali che eseguono Windows. • Aggiunge il nuovo parametro di `WindowsMLRootPath` configurazione che è possibile utilizzare per configurare la cartella dei risultati di inferenza sui dispositivi principali di Windows.

Versione	Modifiche
2.5.4	Nuove funzionalità • Aggiunge il nuovo parametro `UseInstaller` di configurazione che consente di disabilitare lo script di installazione in questo componente.
2.5.3	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.5.2	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.5.1	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.5.0	Versione iniziale.

Adattatore di protocollo Modbus-RTU

Il componente dell'adattatore di protocollo Modbus-RTU (`aws.greengrass.Modbus`) raccoglie informazioni dai dispositivi Modbus RTU locali.

Per richiedere informazioni da un dispositivo Modbus RTU locale con questo componente, pubblica un messaggio sull'argomento a cui questo componente è abbonato. Nel messaggio, specificate la richiesta Modbus RTU da inviare a un dispositivo. Quindi, questo componente pubblica una risposta che contiene il risultato della richiesta Modbus RTU.

> **Note**
>
> Questo componente offre funzionalità simili al connettore dell'adattatore di protocollo Modbus RTU in V1. AWS IoT Greengrass Per ulteriori informazioni, consulta il connettore dell'adattatore di protocollo Modbus RTU nella V1 Developer Guide. AWS IoT Greengrass

Argomenti

- Configurazione
- Dati di input
- Dati di output
- Richieste e risposte RTU Modbus
- File di registro locale
- Licenze
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x
- 2,0x

Type

Questo componente è un componente Lambda () `aws.greengrass.lambda`. Il nucleo Greengrass esegue la funzione Lambda di questo componente utilizzando il componente di avvio Lambda.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato solo sui dispositivi principali di Linux.

Requisiti

Questo componente ha i seguenti requisiti:

- Il dispositivo principale deve soddisfare i requisiti per eseguire le funzioni Lambda. Se desideri che il dispositivo principale esegua funzioni Lambda containerizzate, il dispositivo deve soddisfare i requisiti per farlo. Per ulteriori informazioni, consulta Requisiti della funzione Lambda.
- Python versione 3.7 installata sul dispositivo principale e aggiunta alla variabile di ambiente PATH.
- Una connessione fisica tra il dispositivo AWS IoT Greengrass principale e i dispositivi Modbus. Il dispositivo principale deve essere collegato fisicamente alla rete Modbus RTU tramite una porta seriale, ad esempio una porta USB.

- Per ricevere i dati di output da questo componente, è necessario unire il seguente aggiornamento di configurazione per il componente legacy del router di abbonamento (`aws.greengrass.LegacySubscriptionRouter`) quando si distribuisce questo componente. Questa configurazione specifica l'argomento in cui questo componente pubblica le risposte.

  Legacy subscription router v2.1.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-modbus": {
        "id": "aws-greengrass-modbus",
        "source": "component:aws.greengrass.Modbus",
        "subject": "modbus/adapter/response",
        "target": "cloud"
      }
    }
  }
  ```

  Legacy subscription router v2.0.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-modbus": {
        "id": "aws-greengrass-modbus",
        "source": "arn:aws:lambda:region:aws:function:aws-greengrass-
  modbus:version",
        "subject": "modbus/adapter/response",
        "target": "cloud"
      }
    }
  }
  ```

  - Sostituisci la *`regione`* con quella Regione AWS che usi.
  - Sostituisci la *`versione`* con la versione della funzione Lambda eseguita da questo componente. Per trovare la versione della funzione Lambda, è necessario visualizzare la ricetta per la versione di questo componente che si desidera distribuire. Apri la pagina dei dettagli di questo componente nella AWS IoT Greengrass console e cerca la coppia chiave-valore della funzione Lambda. Questa coppia chiave-valore contiene il nome e la versione della funzione Lambda.

> **Important**
>
> È necessario aggiornare la versione della funzione Lambda sul router di abbonamento legacy ogni volta che si distribuisce questo componente. Ciò garantisce l'utilizzo della versione corretta della funzione Lambda per la versione del componente che si distribuisce.

Per ulteriori informazioni, consulta Creare distribuzione.

- L'adattatore di protocollo Modbus-RTU è supportato per l'esecuzione in un VPC.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.8

La tabella seguente elenca le dipendenze per la versione 2.1.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.4 and 2.1.5

La tabella seguente elenca le dipendenze per le versioni 2.1.4 e 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili

Dipendenza	Versioni compatibili	Tipo di dipendenza
Servizio di scambio di token	^2.0.0	Rigidi

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.8 and 2.1.0

La tabella seguente elenca le dipendenze per le versioni 2.0.8 e 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.7

La tabella seguente elenca le dipendenze per la versione 2.0.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.6

La tabella seguente elenca le dipendenze per la versione 2.0.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.5

La tabella seguente elenca le dipendenze per la versione 2.0.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.4

La tabella seguente elenca le dipendenze per la versione 2.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili

Dipendenza	Versioni compatibili	Tipo di dipendenza
Servizio di scambio di token	^2.0.0	Rigidi

2.0.3

La tabella seguente elenca le dipendenze per la versione 2.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Rigidi
Lanciatore Lambda	>=1.0.0	Rigidi
Runtime Lambda	>=1.0.0	Flessibili
Servizio di scambio di token	>=1.0.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

> **Note**
>
> La configurazione predefinita di questo componente include i parametri della funzione Lambda. Ti consigliamo di modificare solo i seguenti parametri per configurare questo componente sui tuoi dispositivi.

v2.1.x

`lambdaParams`

Un oggetto che contiene i parametri per la funzione Lambda di questo componente. Questo oggetto contiene le seguenti informazioni:

`EnvironmentVariables`

Un oggetto che contiene i parametri della funzione Lambda. Questo oggetto contiene le seguenti informazioni:

`ModbusLocalPort`

Il percorso assoluto verso la porta seriale Modbus fisica sul dispositivo principale, ad esempio`/dev/ttyS2`.

Per eseguire questo componente in un contenitore, è necessario definire questo percorso come dispositivo di sistema (in`containerParams.devices`) a cui il componente può accedere. Per impostazione predefinita, questo componente viene eseguito in un contenitore.

> **Note**
>
> Questo componente deve avere accesso in lettura/scrittura al dispositivo.

`ModbusBaudRate`

(Facoltativo) Un valore di stringa che specifica la velocità di trasmissione per la comunicazione seriale con i dispositivi Modbus TCP locali.

Impostazione predefinita: `9600`

`ModbusByteSize`

(Facoltativo) Un valore di stringa che specifica la dimensione di un byte nella comunicazione seriale con dispositivi Modbus TCP locali. Scegliete`5`,`6`, o bit`7`. `8`

Impostazione predefinita: `8`

`ModbusParity`

(Facoltativo) La modalità di parità da utilizzare per verificare l'integrità dei dati nella comunicazione seriale con dispositivi Modbus TCP locali.

- `E`— Verifica l'integrità dei dati con parità uniforme.
- `0`— Verifica l'integrità dei dati con parità dispari.

- `N`— Non verificate l'integrità dei dati.

Impostazione predefinita: `N`

`ModbusStopBits`

(Facoltativo) Un valore di stringa che specifica il numero di bit che indicano la fine di un byte nella comunicazione seriale con dispositivi Modbus TCP locali.

Impostazione predefinita: `1`

`containerMode`

(Facoltativo) La modalità di containerizzazione per questo componente. Seleziona una delle opzioni seguenti:

- `GreengrassContainer`— Il componente viene eseguito in un ambiente di runtime isolato all'interno del AWS IoT Greengrass contenitore.

  Se si specifica questa opzione, è necessario specificare un dispositivo di sistema (in`containerParams.devices`) per consentire al contenitore di accedere al dispositivo Modbus.
- `NoContainer`— Il componente non viene eseguito in un ambiente di runtime isolato.

Impostazione predefinita: `GreengrassContainer`

`containerParams`

(Facoltativo) Un oggetto che contiene i parametri del contenitore per questo componente. Il componente utilizza questi parametri se si specifica `GreengrassContainer` per`containerMode`.

Questo oggetto contiene le seguenti informazioni:

`memorySize`

(Facoltativo) La quantità di memoria (in kilobyte) da allocare al componente.

Il valore predefinito è 512 MB (525.312 KB).

`devices`

(Facoltativo) Un oggetto che specifica i dispositivi di sistema a cui il componente può accedere in un contenitore.

> **Important**
>
> Per eseguire questo componente in un contenitore, è necessario specificare il dispositivo di sistema configurato nella variabile di `ModbusLocalPort` ambiente.

Questo oggetto contiene le seguenti informazioni:

`0`— Si tratta di un indice di matrice sotto forma di stringa.

Un oggetto che contiene le seguenti informazioni:

`path`

Il percorso del dispositivo di sistema sul dispositivo principale. Deve avere lo stesso valore del valore per cui è stato configurato`ModbusLocalPort`.

`permission`

(Facoltativo) L'autorizzazione ad accedere al dispositivo di sistema dal contenitore. Questo valore deve essere`rw`, che specifica che il componente ha accesso in lettura/scrittura al dispositivo di sistema.

Impostazione predefinita: `rw`

`addGroupOwner`

(Facoltativo) Se aggiungere o meno il gruppo di sistema che esegue il componente come proprietario del dispositivo di sistema.

Impostazione predefinita: `true`

`pubsubTopics`

(Facoltativo) Un oggetto che contiene gli argomenti a cui il componente si iscrive per ricevere messaggi. È possibile specificare ogni argomento e se il componente sottoscrive gli argomenti MQTT AWS IoT Core o gli argomenti di pubblicazione/sottoscrizione locali.

Questo oggetto contiene le seguenti informazioni:

`0`— Si tratta di un indice di matrice sotto forma di stringa.

Un oggetto che contiene le seguenti informazioni:

`type`

(Facoltativo) Il tipo di messaggi di pubblicazione/sottoscrizione utilizzato da questo componente per sottoscrivere i messaggi. Seleziona una delle opzioni seguenti:

- `PUB_SUB`: iscriviti ai messaggi di pubblicazione/sottoscrizione locali. Se scegli questa opzione, l'argomento non può contenere caratteri jolly MQTT. Per ulteriori informazioni su come inviare messaggi dal componente personalizzato quando si specifica questa opzione, vedere. Pubblicare/sottoscrivere messaggi locali
- `IOT_CORE`— Abbonarsi ai messaggi AWS IoT Core MQTT. Se scegli questa opzione, l'argomento può contenere caratteri jolly MQTT. Per ulteriori informazioni su come inviare messaggi da componenti personalizzati quando si specifica questa opzione, vedere. AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

Impostazione predefinita: `PUB_SUB`

`topic`

(Facoltativo) L'argomento a cui il componente si iscrive per ricevere messaggi. Se si specifica `IotCore` for`type`, è possibile utilizzare i caratteri jolly MQTT (+and#) in questo argomento.

Example Esempio: aggiornamento basato sull'unione della configurazione (modalità contenitore)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "GreengrassContainer",
  "containerParams": {
    "devices": {
      "0": {
        "path": "/dev/ttyS2",
        "permission": "rw",
        "addGroupOwner": true
      }
    }
  }
}
```

Example Esempio: aggiornamento tramite fusione della configurazione (nessuna modalità contenitore)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "NoContainer"
}
```

v2.0.x

`lambdaParams`

Un oggetto che contiene i parametri per la funzione Lambda di questo componente. Questo oggetto contiene le seguenti informazioni:

`EnvironmentVariables`

Un oggetto che contiene i parametri della funzione Lambda. Questo oggetto contiene le seguenti informazioni:

`ModbusLocalPort`

Il percorso assoluto verso la porta seriale Modbus fisica sul dispositivo principale, ad esempio`/dev/ttyS2`.

Per eseguire questo componente in un contenitore, è necessario definire questo percorso come dispositivo di sistema (in`containerParams.devices`) a cui il componente può accedere. Per impostazione predefinita, questo componente viene eseguito in un contenitore.

> **Note**
>
> Questo componente deve avere accesso in lettura/scrittura al dispositivo.

`containerMode`

(Facoltativo) La modalità di containerizzazione per questo componente. Seleziona una delle opzioni seguenti:

- `GreengrassContainer`— Il componente viene eseguito in un ambiente di runtime isolato all'interno del AWS IoT Greengrass contenitore.

  Se si specifica questa opzione, è necessario specificare un dispositivo di sistema (in`containerParams.devices`) per consentire al contenitore di accedere al dispositivo Modbus.
- `NoContainer`— Il componente non viene eseguito in un ambiente di runtime isolato.

Impostazione predefinita: `GreengrassContainer`

`containerParams`

(Facoltativo) Un oggetto che contiene i parametri del contenitore per questo componente. Il componente utilizza questi parametri se si specifica `GreengrassContainer` per`containerMode`.

Questo oggetto contiene le seguenti informazioni:

`memorySize`

(Facoltativo) La quantità di memoria (in kilobyte) da allocare al componente.

Il valore predefinito è 512 MB (525.312 KB).

`devices`

(Facoltativo) Un oggetto che specifica i dispositivi di sistema a cui il componente può accedere in un contenitore.

> **Important**
>
> Per eseguire questo componente in un contenitore, è necessario specificare il dispositivo di sistema configurato nella variabile di `ModbusLocalPort` ambiente.

Questo oggetto contiene le seguenti informazioni:

`0`— Si tratta di un indice di matrice sotto forma di stringa.

Un oggetto che contiene le seguenti informazioni:

`path`

Il percorso del dispositivo di sistema sul dispositivo principale. Deve avere lo stesso valore del valore per cui è stato configuratoModbusLocalPort.

`permission`

(Facoltativo) L'autorizzazione ad accedere al dispositivo di sistema dal contenitore. Questo valore deve essere`rw`, che specifica che il componente ha accesso in lettura/scrittura al dispositivo di sistema.

Impostazione predefinita: `rw`

`addGroupOwner`

(Facoltativo) Se aggiungere o meno il gruppo di sistema che esegue il componente come proprietario del dispositivo di sistema.

Impostazione predefinita: `true`

`pubsubTopics`

(Facoltativo) Un oggetto che contiene gli argomenti a cui il componente si iscrive per ricevere messaggi. È possibile specificare ogni argomento e se il componente sottoscrive gli argomenti MQTT AWS IoT Core o gli argomenti di pubblicazione/sottoscrizione locali.

Questo oggetto contiene le seguenti informazioni:

`0`— Si tratta di un indice di matrice sotto forma di stringa.

Un oggetto che contiene le seguenti informazioni:

`type`

(Facoltativo) Il tipo di messaggi di pubblicazione/sottoscrizione utilizzato da questo componente per sottoscrivere i messaggi. Seleziona una delle opzioni seguenti:

- `PUB_SUB`: iscriviti ai messaggi di pubblicazione/sottoscrizione locali. Se scegli questa opzione, l'argomento non può contenere caratteri jolly MQTT. Per ulteriori informazioni su come inviare messaggi dal componente personalizzato quando si specifica questa opzione, vedere. Pubblicare/sottoscrivere messaggi locali
- `IOT_CORE`— Abbonarsi ai messaggi AWS IoT Core MQTT. Se scegli questa opzione, l'argomento può contenere caratteri jolly MQTT. Per ulteriori informazioni su come inviare messaggi da componenti personalizzati quando si specifica questa opzione, vedere. AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

Impostazione predefinita: `PUB_SUB`

`topic`

(Facoltativo) L'argomento a cui il componente si iscrive per ricevere messaggi. Se si specifica `IotCore` for`type`, è possibile utilizzare i caratteri jolly MQTT (+and#) in questo argomento.

Example Esempio: aggiornamento basato sull'unione della configurazione (modalità contenitore)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "GreengrassContainer",
  "containerParams": {
    "devices": {
      "0": {
        "path": "/dev/ttyS2",
        "permission": "rw",
        "addGroupOwner": true
      }
    }
  }
}
```

Example Esempio: aggiornamento tramite fusione della configurazione (nessuna modalità contenitore)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "ModbusLocalPort": "/dev/ttyS2"
    }
  },
  "containerMode": "NoContainer"
}
```

Dati di input

Questo componente accetta i parametri di richiesta Modbus RTU sul seguente argomento e invia la richiesta Modbus RTU al dispositivo. Per impostazione predefinita, questo componente sottoscrive i messaggi di pubblicazione/sottoscrizione locali. Per ulteriori informazioni su come pubblicare messaggi su questo componente dai componenti personalizzati, consulta. Pubblicare/sottoscrivere messaggi locali

Argomento predefinito (pubblicazione/sottoscrizione locale): `modbus/adapter/request`

Il messaggio accetta le seguenti proprietà. I messaggi di input devono essere in formato JSON.

`request`

I parametri per la richiesta Modbus RTU da inviare.

La forma del messaggio di richiesta dipende dal tipo di richiesta Modbus RTU che rappresenta. Le seguenti proprietà sono obbligatorie per tutte le richieste.

Tipo: `object` che contiene le seguenti informazioni:

`operation`

Il nome dell'operazione da eseguire. Ad esempio, specificate `ReadCoilsRequest` di leggere le bobine su un dispositivo Modbus RTU. Per ulteriori informazioni sulle operazioni supportate, vedere. Richieste e risposte RTU Modbus

Tipo: `string`

`device`

Il dispositivo di destinazione della richiesta.

Questo valore deve essere un numero intero compreso tra `0` e `247`.

Tipo: `integer`

Gli altri parametri di includere nella richiesta variano a seconda dell'operazione. Questo componente gestisce il controllo di ridondanza ciclico (CRC) per verificare le richieste di dati per te.

> **Note**
>
> Se la richiesta include una `address` proprietà, è necessario specificarne il valore come numero intero. Ad esempio, `"address": 1`.

`id`

Un ID arbitrario della richiesta. Utilizzate questa proprietà per mappare una richiesta di input a una risposta di output. Quando specificate questa proprietà, il componente imposta la `id` proprietà nell'oggetto di risposta su questo valore.

Tipo: `string`

Example Esempio di input: richiesta lettura nastri

```
{
  "request": {
    "operation": "ReadCoilsRequest",
    "device": 1,
    "address": 1,
    "count": 1
  },
  "id": "MyRequest"
}
```

Dati di output

Per impostazione predefinita, questo componente pubblica le risposte come dati di output sul seguente argomento MQTT. È necessario specificare questo argomento come contenuto `subject` nella configurazione del componente legacy del router di abbonamento. Per ulteriori informazioni su come sottoscrivere i messaggi relativi a questo argomento nei componenti personalizzati, consultaAWS IoT Core Pubblicare/sottoscrivere messaggi MQTT.

Argomento predefinito (AWS IoT Core MQTT): `modbus/adapter/response`

La forma del messaggio di risposta dipende dall'operazione di richiesta e dallo stato della risposta. Per alcuni esempi, consulta Richieste e risposte di esempio.

Ogni risposta include le seguenti proprietà:

`response`

La risposta del dispositivo Modbus RTU.

Tipo: `object` che contiene le seguenti informazioni:

`status`

Stato della richiesta. Lo stato può avere uno dei seguenti valori:

- `Success`— La richiesta era valida, il componente ha inviato la richiesta alla rete Modbus RTU e la rete Modbus RTU ha restituito una risposta.
- `Exception`— La richiesta era valida, il componente ha inviato la richiesta alla rete Modbus RTU e la rete Modbus RTU ha restituito un'eccezione. Per ulteriori informazioni, consulta Stato risposta: eccezione.
- `No Response`— La richiesta non era valida e il componente ha rilevato l'errore prima di inviarla alla rete Modbus RTU. Per ulteriori informazioni, consulta Stato risposta: nessuna risposta.

`operation`

Operazione richiesta dal componente.

`device`

Il dispositivo a cui il componente ha inviato la richiesta.

`payload`

La risposta del dispositivo Modbus RTU. Se `status` è`No Response`, questo oggetto contiene solo una `error` proprietà con la descrizione dell'errore (ad esempio,`[Input/Output] No Response received from the remote unit`).

`id`

L'ID della richiesta, che è possibile utilizzare per identificare quale risposta corrisponde a quale richiesta.

> **Note**
>
> Una risposta a un'operazione di scrittura è semplicemente un eco della richiesta. Sebbene le risposte di scrittura non includano informazioni significative, è buona norma controllare lo stato della risposta per vedere se la richiesta ha esito positivo o negativo.

Example Output di esempio: Operazione riuscita

```
{
  "response" : {
    "status" : "success",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "function_code": 1,
      "bits": [1]
    }
  },
  "id" : "MyRequest"
}
```

Example Esempio di output: Errore

```
{
  "response" : {
    "status" : "fail",
    "error_message": "Internal Error",
    "error": "Exception",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "function_code": 129,
      "exception_code": 2
    }
  },
  "id" : "MyRequest"
}
```

Per ulteriori esempi, consulta Richieste e risposte di esempio.

Richieste e risposte RTU Modbus

Questo connettore accetta i parametri della richiesta RTU Modbus come dati di input e pubblica le risposte come dati di output.

Sono supportate le seguenti operazioni comuni.

Nome dell'operazione nella richiesta	Codice della funzione in risposta
ReadCoilsRequest	01
ReadDiscreteInputsRequest	02
ReadHoldingRegistersRequest	03
ReadInputRegistersRequest	04
WriteSingleCoilRequest	05
WriteSingleRegisterRequest	06
WriteMultipleCoilsRequest	15
WriteMultipleRegistersRequest	16
MaskWriteRegisterRequest	22
ReadWriteMultipleRegistersRequest	23

Richieste e risposte di esempio

Di seguito sono riportate alcune richieste e risposte di esempio per le operazioni supportate.

Leggi le bobine

Esempio di richiesta:

```
{
  "request": {
    "operation": "ReadCoilsRequest",
    "device": 1,
    "address": 1,
    "count": 1
  },
  "id": "TestRequest"
}
```

Esempio di risposta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "function_code": 1,
      "bits": [1]
    }
  },
  "id" : "TestRequest"
}
```

Leggi gli ingressi discreti

Esempio di richiesta:

```
{
  "request": {
    "operation": "ReadDiscreteInputsRequest",
    "device": 1,
    "address": 1,
    "count": 1
  },
  "id": "TestRequest"
}
```

Esempio di risposta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "ReadDiscreteInputsRequest",
    "payload": {
      "function_code": 2,
      "bits": [1]
    }
  },
  "id" : "TestRequest"
}
```

Leggi i registri di detenzione

Esempio di richiesta:

```
{
  "request": {
    "operation": "ReadHoldingRegistersRequest",
    "device": 1,
    "address": 1,
    "count": 1
  },
  "id": "TestRequest"
}
```

Esempio di risposta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "ReadHoldingRegistersRequest",
    "payload": {
      "function_code": 3,
      "registers": [20,30]
    }
  },
  "id" : "TestRequest"
}
```

Leggi i registri di input

Esempio di richiesta:

```
{
  "request": {
    "operation": "ReadInputRegistersRequest",
    "device": 1,
    "address": 1,
    "count": 1
  },
  "id": "TestRequest"
}
```

Scrivi bobina singola

Esempio di richiesta:

```
{
  "request": {
    "operation": "WriteSingleCoilRequest",
    "device": 1,
    "address": 1,
    "value": 1
  },
  "id": "TestRequest"
}
```

Esempio di risposta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "WriteSingleCoilRequest",
    "payload": {
      "function_code": 5,
      "address": 1,
      "value": true
    }
  },
  "id" : "TestRequest"
}
```

Scrivi un registro singolo

Esempio di richiesta:

```
{
  "request": {
    "operation": "WriteSingleRegisterRequest",
    "device": 1,
    "address": 1,
    "value": 1
  },
  "id": "TestRequest"
```

```
}
```

Scrivi più bobine

Esempio di richiesta:

```
{
  "request": {
    "operation": "WriteMultipleCoilsRequest",
    "device": 1,
    "address": 1,
    "values": [1,0,0,1]
  },
  "id": "TestRequest"
}
```

Esempio di risposta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "WriteMultipleCoilsRequest",
    "payload": {
      "function_code": 15,
      "address": 1,
      "count": 4
    }
  },
  "id" : "TestRequest"
}
```

Scrivi più registri

Esempio di richiesta:

```
{
  "request": {
    "operation": "WriteMultipleRegistersRequest",
    "device": 1,
    "address": 1,
    "values": [20,30,10]
  },
```

```
  "id": "TestRequest"
}
```

Esempio di risposta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "WriteMultipleRegistersRequest",
    "payload": {
      "function_code": 23,
      "address": 1,
      "count": 3
    }
  },
  "id" : "TestRequest"
}
```

Maschera, scrittura, registro

Esempio di richiesta:

```
{
  "request": {
    "operation": "MaskWriteRegisterRequest",
    "device": 1,
    "address": 1,
    "and_mask": 175,
    "or_mask": 1
  },
  "id": "TestRequest"
}
```

Esempio di risposta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "MaskWriteRegisterRequest",
    "payload": {
      "function_code": 22,
```

```
      "and_mask": 0,
      "or_mask": 8
    }
  },
  "id" : "TestRequest"
}
```

Leggi e scrivi più registri

Esempio di richiesta:

```
{
  "request": {
    "operation": "ReadWriteMultipleRegistersRequest",
    "device": 1,
    "read_address": 1,
    "read_count": 2,
    "write_address": 3,
    "write_registers": [20,30,40]
  },
  "id": "TestRequest"
}
```

Esempio di risposta:

```
{
  "response": {
    "status": "success",
    "device": 1,
    "operation": "ReadWriteMultipleRegistersRequest",
    "payload": {
      "function_code": 23,
      "registers": [10,20,10,20]
    }
  },
  "id" : "TestRequest"
}
```

> **Note**
>
> La risposta include i registri letti dal componente.

Stato risposta: eccezione

Le eccezioni possono verificarsi se il formato della richiesta è valido, ma la richiesta non è stata completata. In questo caso, la risposta contiene le seguenti informazioni:

- Il `status` è impostato su `Exception`.
- `function_code` è pari al codice della funzione della richiesta + 128.
- `exception_code` contiene il codice dell'eccezione. Per ulteriori informazioni sull'eccezione , consulta Codici delle eccezioni Modbus.

Esempio:

```
{
  "response": {
    "status": "fail",
    "error_message": "Internal Error",
    "error": "Exception",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "function_code": 129,
      "exception_code": 2
    }
  },
  "id": "TestRequest"
}
```

Stato risposta: nessuna risposta

Questo connettore esegue controlli di convalida sulla richiesta Modbus. Ad esempio, verifica l'eventuale presenza di formati non validi e campi non compilati. Se la convalida ha esito negativo, il connettore non invia la richiesta. Al contrario, restituirà una risposta contenente le seguenti informazioni:

- Il `status` è impostato su `No Response`.
- `error` contiene il motivo dell'errore.
- `error_message` contiene il messaggio dell'errore.

Esempi:

```
{
  "response": {
    "status": "fail",
    "error_message": "Invalid address field. Expected <type 'int'>, got <type 'str'>",
    "error": "No Response",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "error": "Invalid address field. Expected Expected <type 'int'>, got <type
 'str'>"
    }
  },
  "id": "TestRequest"
}
```

Se la richiesta è destinata a un dispositivo inesistente o se la rete RTU Modbus non funziona, potrebbe venire restituito `ModbusIOException`, che utilizza il formato Nessuna risposta.

```
{
  "response": {
    "status": "fail",
    "error_message": "[Input/Output] No Response received from the remote unit",
    "error": "No Response",
    "device": 1,
    "operation": "ReadCoilsRequest",
    "payload": {
      "error": "[Input/Output] No Response received from the remote unit"
    }
  },
  "id": "TestRequest"
}
```

File di registro locale

Questo componente utilizza il seguente file di registro.

```
/greengrass/v2/logs/aws.greengrass.Modbus.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale.

```
sudo tail -f /greengrass/v2/logs/aws.greengrass.Modbus.log
```

Licenze

Questo componente include i seguenti software/licenze di terze parti:

- Licenza pymodbus/BSD
- Licenza pyserial /BSD

Questo componente è rilasciato in base al contratto di licenza del software Greengrass Core.

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
2.1.8	Versione aggiornata per Greengrass nucleus versione 2.12.0.
2.1.7	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.5	Correzioni di bug e miglioramenti • Risolve un problema con l'`ReadDiscreteInput` operazione.
2.1.4	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per Greengrass nucleus versione 2.7.0.
2.1.1	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.

Versione	Modifiche
2.1.0	Nuove funzionalità • Aggiunge le `ModbusStopBits` opzioni`ModbusBaudRate`, `ModbusByteSize` `ModbusParity`, e che è possibile specificare per configurare la comunicazione seriale con i dispositivi Modbus RTU.
2.0.8	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.0.7	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.6	Versione aggiornata per Greengrass nucleus versione 2.3.0.
2.0.5	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.0.4	Versione aggiornata per la versione 2.1.0 di Greengrass nucleus.
2.0.3	Versione iniziale.

Ponte MQTT

Il componente bridge MQTT (`aws.greengrass.clientdevices.mqtt.Bridge`) inoltra i messaggi MQTT tra i dispositivi client, il servizio di pubblicazione/sottoscrizione locale di Greengrass e. AWS IoT Core È possibile utilizzare questo componente per agire sui messaggi MQTT provenienti dai dispositivi client in componenti personalizzati e sincronizzare i dispositivi client con. Cloud AWS

> **Note**
>
> I dispositivi client sono dispositivi IoT locali che si connettono a un dispositivo core Greengrass per inviare messaggi MQTT e dati da elaborare. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

È possibile utilizzare questo componente per inoltrare messaggi tra i seguenti broker di messaggi:

- MQTT locale: il broker MQTT locale gestisce i messaggi tra i dispositivi client e un dispositivo principale.

- Pubblicazione/sottoscrizione locale: il broker di messaggi Greengrass locale gestisce i messaggi tra i componenti di un dispositivo principale. Per ulteriori informazioni su come interagire con questi messaggi nei componenti Greengrass, vedere. Pubblicare/sottoscrivere messaggi locali
- AWS IoT Core— Il broker AWS IoT Core MQTT gestisce i messaggi tra dispositivi IoT e Cloud AWS destinazioni. Per ulteriori informazioni su come interagire con questi messaggi nei componenti Greengrass, vedere. AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.3.x
- 2.2.x
- 2.1.x

- 2,0x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Se si configura il componente broker MQTT del dispositivo principale per utilizzare una porta diversa dalla porta predefinita 8883, è necessario utilizzare MQTT bridge v2.1.0 o versione successiva. Configuralo per connetterti alla porta in cui opera il broker.
- Il componente bridge MQTT è supportato per l'esecuzione in un VPC.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le

dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.3.0

La tabella seguente elenca le dipendenze per la versione 2.3.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.2.0 <2.5.0	Rigidi

2.2.5 and 2.2.6

La tabella seguente elenca le dipendenze per le versioni 2.2.5 e 2.2.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.2.0 <2.5.0	Rigidi

2.2.3 and 2.2.4

La tabella seguente elenca le dipendenze per le versioni 2.2.3 e 2.2.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.2.0 <2.4.0	Rigidi

2.2.0 – 2.2.2

La tabella seguente elenca le dipendenze per le versioni da 2.2.0 a 2.2.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.2.0 <2.3.0	Rigidi

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.0.0 <2.2.0	Rigidi

2.0.0 to 2.1.0

La tabella seguente elenca le dipendenze per le versioni da 2.0.0 a 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.0.0 <2.1.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

2.3.0

`mqttTopicMapping`

Le mappature degli argomenti che desideri collegare. Questo componente sottoscrive i messaggi sull'argomento di origine e pubblica i messaggi che riceve nell'argomento di

destinazione. Ogni mappatura degli argomenti definisce l'argomento, il tipo di origine e il tipo di destinazione.

Questo oggetto contiene le seguenti informazioni:

topicMappingNameKey

Il nome di questa mappatura degli argomenti. Sostituisci *topicMappingNameKey* con un nome che ti aiuti a identificare questa mappatura degli argomenti.

Questo oggetto contiene le seguenti informazioni:

`topic`

L'argomento o il filtro degli argomenti da collegare tra il broker di origine e quello di destinazione.

È possibile utilizzare i caratteri jolly degli argomenti + e # MQTT per inoltrare messaggi su tutti gli argomenti che corrispondono a un filtro per argomento. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide. AWS IoT Core

> **Note**
>
> Per utilizzare i caratteri jolly degli argomenti MQTT con il broker dei `Pubsub` sorgenti, è necessario utilizzare la versione 2.6.0 o successiva del componente Greengrass nucleus.

`targetTopicPrefix`

Il prefisso da aggiungere all'argomento di destinazione quando questo componente inoltra il messaggio.

`source`

Il broker di messaggi di origine. Seleziona una delle opzioni seguenti:

- `LocalMqtt`— Il broker MQTT locale con cui comunicano i dispositivi client.
- `Pubsub`— Il broker di messaggi di pubblicazione/sottoscrizione locale Greengrass.
- `IotCore`— Il AWS IoT Core broker di messaggi MQTT.

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

`source` e `target` deve essere diverso.

`target`

Il broker di messaggi di destinazione. Seleziona una delle opzioni seguenti:

- `LocalMqtt`— Il broker MQTT locale con cui comunicano i dispositivi client.
- `Pubsub`— Il broker di messaggi di pubblicazione/sottoscrizione locale Greengrass.
- `IotCore`— Il AWS IoT Core broker di messaggi MQTT.

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

`source` e `target` deve essere diverso.

mqtt5 RouteOptions

(Facoltativo) Fornisce opzioni per configurare le mappature degli argomenti per collegare i messaggi dall'argomento di origine a quello di destinazione.

Questo oggetto contiene le seguenti informazioni:

mqtt5 RouteOptionsNameKey

Il nome delle opzioni di percorso per la mappatura di un argomento. Sostituisci *mqtt5 RouteOptionsNameKey* con la *topicMappingNamechiave* corrispondente definita nel campo. `mqttTopicMapping`

Questo oggetto contiene le seguenti informazioni:

NoLocal

(Facoltativo) Se abilitato, il bridge non inoltra messaggi su un argomento pubblicato dal bridge stesso. Utilizzatelo per prevenire i loop, come segue:

```
{
    "mqtt5RouteOptions": {
        "toIoTCore": {
            "noLocal": true
        }
    },
    "mqttTopicMapping": {
        "toIoTCore": {
            "topic": "device",
            "source": "LocalMqtt",
            "target": "IotCore"
        },
        "toLocal": {
            "topic": "device",
            "source": "IotCore",
            "target": "LocalMqtt"
        }
    }
}
```

`noLocal`è supportato solo per i percorsi in cui `source` è`LocalMqtt`.

Impostazione predefinita: false

retainAsPublished

(Facoltativo) Se abilitato, i messaggi inoltrati dal bridge hanno lo stesso `retain` contrassegno dei messaggi pubblicati sul broker per quella rotta.

`retainAsPublished`è supportato solo per i percorsi in cui è`source`. `LocalMqtt`

Impostazione predefinita: false

mqtt

(Facoltativo) Impostazioni del protocollo MQTT per la comunicazione con il broker locale.

version

(Facoltativo) La versione del protocollo MQTT utilizzata dal bridge per comunicare con il broker locale. Deve essere la stessa della versione MQTT selezionata nella configurazione del nucleo.

Scegli tra le seguenti opzioni:

- `mqtt3`
- `mqtt5`

È necessario implementare un broker MQTT quando il `target` campo `source` o dell'`mqttTopicMapping`oggetto è impostato su. `LocalMqtt` Se si sceglie l'`mqtt5`opzione, è necessario utilizzare il. Broker MQTT 5 (EMQX)

Impostazione predefinita: `mqtt3`

ackTimeoutSeconds

(Facoltativo) Intervallo di tempo in cui attendere i pacchetti PUBACK, SUBACK o UNSUBACK prima di fallire l'operazione.

Impostazione predefinita: 60

connAckTimeoutSig.ra

(Facoltativo) Intervallo di tempo di attesa di un pacchetto CONNACK prima di interrompere la connessione.

Impostazione predefinita: 20000 (20 secondi)

pingTimeoutMs

(Facoltativo) La quantità di tempo in millisecondi che il bridge attende per ricevere un messaggio PINGACK dal broker locale. Se l'attesa supera il timeout, il bridge si chiude e riapre la connessione MQTT. Questo valore deve essere inferiore a. `keepAliveTimeoutSeconds`

Impostazione predefinita: 30000 (30 secondi)

keepAliveTimeoutSecondi

(Facoltativo) La quantità di tempo in secondi tra ogni messaggio PING inviato dal bridge per mantenere attiva la connessione MQTT. Questo valore deve essere maggiore di. `pingTimeoutMs`

Impostazione predefinita: 60

maxReconnectDelaySig.ra

(Facoltativo) Il tempo massimo in secondi per la riconnessione di MQTT.

Impostazione predefinita: 30000 (30 secondi)

minReconnectDelaySig.ra

(Facoltativo) Il tempo minimo in secondi per la riconnessione di MQTT.

Ricevi il massimo

(Facoltativo) Il numero massimo di pacchetti QoS1 non riconosciuti che il bridge può inviare.

Impostazione predefinita: 100

maximumPacketSize

Il numero massimo di byte che il client accetterà per un pacchetto MQTT.

Impostazione predefinita: null (nessun limite)

sessionExpiryInterval

(Facoltativo) La quantità di tempo in secondi che è possibile richiedere per la durata di una sessione tra il bridge e il broker locale.

Impostazione predefinita: 4294967295 (la sessione non scade mai)

`brokerUri`

(Facoltativo) L'URI del broker MQTT locale. È necessario specificare questo parametro se si configura il broker MQTT per utilizzare una porta diversa dalla porta predefinita 8883. Utilizzate il seguente formato e sostituite la *`porta`* con la quale opera il broker MQTT:. `ssl://localhost:`*`port`*

Impostazione predefinita: `ssl://localhost:8883`

startupTimeoutSeconds

(Facoltativo) Il tempo massimo in secondi per l'avvio del componente. Lo stato del componente cambia `BROKEN` se supera questo timeout.

Impostazione predefinita: `120`

Example Esempio: fusione e aggiornamento della configurazione

Il seguente esempio di aggiornamento della configurazione specifica quanto segue:

- Inoltra i messaggi dai dispositivi client agli AWS IoT Core argomenti che corrispondono al filtro degli `clients/+/hello/world` argomenti.
- Inoltra i messaggi dai dispositivi client alla pubblicazione o sottoscrizione locale su argomenti che corrispondono al filtro degli argomenti e aggiungi il `events/input/` prefisso all'`clients/+/detections`argomento di destinazione. L'argomento di destinazione risultante corrisponde al filtro degli argomenti. `events/input/clients/+/detections`
- Inoltra i messaggi dai dispositivi client AWS IoT Core agli argomenti che corrispondono al filtro degli argomenti e aggiungi il `$aws/rules/StatusUpdateRule/` prefisso all'`clients/+/status`argomento di destinazione. Questo esempio inoltra questi messaggi direttamente a una AWS IoTregola denominata per `StatusUpdateRule` ridurre i costi utilizzando Basic Ingest.

```
{
  "mqttTopicMapping": {
    "ClientDeviceHelloWorld": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "ClientDeviceEvents": {
      "topic": "clients/+/detections",
      "targetTopicPrefix": "events/input/",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ClientDeviceCloudStatusUpdate": {
      "topic": "clients/+/status",
      "targetTopicPrefix": "$aws/rules/StatusUpdateRule/",
      "source": "LocalMqtt",
      "target": "IotCore"
```

```
    }
  }
}
```

Example Esempio: configurazione di MQTT 5

La seguente configurazione di esempio aggiorna quanto segue:

- Consente al bridge di utilizzare il protocollo MQTT 5 con il broker locale.
- Configura MQTT keep come impostazione pubblicata per la mappatura degli `ClientDeviceHelloWorld` argomenti.

```
{
  "mqttTopicMapping": {
    "ClientDeviceHelloWorld": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  },
  "mqtt5RouteOptions": {
    "ClientDeviceHelloWorld": {
      "retainAsPublished": true
    }
  },
  "mqtt": {
    "version": "mqtt5"
  }
}
```

2.2.6

`mqttTopicMapping`

Le mappature degli argomenti che desideri collegare. Questo componente sottoscrive i messaggi sull'argomento di origine e pubblica i messaggi che riceve nell'argomento di destinazione. Ogni mappatura degli argomenti definisce l'argomento, il tipo di origine e il tipo di destinazione.

Questo oggetto contiene le seguenti informazioni:

topicMappingNameKey

Il nome di questa mappatura degli argomenti. Sostituisci *topicMappingNameKey* con un nome che ti aiuti a identificare questa mappatura degli argomenti.

Questo oggetto contiene le seguenti informazioni:

`topic`

L'argomento o il filtro degli argomenti da collegare tra il broker di origine e quello di destinazione.

È possibile utilizzare i caratteri jolly degli argomenti + e # MQTT per inoltrare messaggi su tutti gli argomenti che corrispondono a un filtro per argomento. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide. AWS IoT Core

> **Note**
>
> Per utilizzare i caratteri jolly degli argomenti MQTT con il broker dei `Pubsub` sorgenti, è necessario utilizzare la versione 2.6.0 o successiva del componente Greengrass nucleus.

`targetTopicPrefix`

Il prefisso da aggiungere all'argomento di destinazione quando questo componente inoltra il messaggio.

`source`

Il broker di messaggi di origine. Seleziona una delle opzioni seguenti:

- `LocalMqtt`— Il broker MQTT locale con cui comunicano i dispositivi client.
- `Pubsub`— Il broker di messaggi di pubblicazione/sottoscrizione locale Greengrass.
- `IotCore`— Il AWS IoT Core broker di messaggi MQTT.

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare

una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

`source` e `target` deve essere diverso.

`target`

Il broker di messaggi di destinazione. Seleziona una delle opzioni seguenti:

- `LocalMqtt`— Il broker MQTT locale con cui comunicano i dispositivi client.
- `Pubsub`— Il broker di messaggi di pubblicazione/sottoscrizione locale Greengrass.
- `IotCore`— Il AWS IoT Core broker di messaggi MQTT.

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

`source` e `target` deve essere diverso.

`brokerUri`

(Facoltativo) L'URI del broker MQTT locale. È necessario specificare questo parametro se si configura il broker MQTT per utilizzare una porta diversa dalla porta predefinita 8883. Utilizzate il seguente formato e sostituite la *`porta`* con la quale opera il broker MQTT:. `ssl://localhost:`*`port`*

Impostazione predefinita: `ssl://localhost:8883`

startupTimeoutSeconds

(Facoltativo) Il tempo massimo in secondi per l'avvio del componente. Lo stato del componente cambia `BROKEN` se supera questo timeout.

Impostazione predefinita: `120`

Example Esempio: fusione e aggiornamento della configurazione

Il seguente esempio di aggiornamento della configurazione specifica quanto segue:

- Inoltra i messaggi dai dispositivi client agli AWS IoT Core argomenti che corrispondono al filtro degli `clients/+/hello/world` argomenti.
- Inoltra i messaggi dai dispositivi client alla pubblicazione o sottoscrizione locale su argomenti che corrispondono al filtro degli argomenti e aggiungi il `events/input/` prefisso all'`clients/+/detections`argomento di destinazione. L'argomento di destinazione risultante corrisponde al filtro degli argomenti. `events/input/clients/+/detections`
- Inoltra i messaggi dai dispositivi client AWS IoT Core agli argomenti che corrispondono al filtro degli argomenti e aggiungi il `$aws/rules/StatusUpdateRule/` prefisso all'`clients/+/status`argomento di destinazione. Questo esempio inoltra questi messaggi direttamente a una AWS IoTregola denominata per `StatusUpdateRule` ridurre i costi utilizzando Basic Ingest.

```
{
  "mqttTopicMapping": {
    "ClientDeviceHelloWorld": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "ClientDeviceEvents": {
      "topic": "clients/+/detections",
      "targetTopicPrefix": "events/input/",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ClientDeviceCloudStatusUpdate": {
      "topic": "clients/+/status",
      "targetTopicPrefix": "$aws/rules/StatusUpdateRule/",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

2.2.0 - 2.2.5

`mqttTopicMapping`

Le mappature degli argomenti che vuoi collegare. Questo componente sottoscrive i messaggi sull'argomento di origine e pubblica i messaggi che riceve nell'argomento di destinazione. Ogni mappatura degli argomenti definisce l'argomento, il tipo di origine e il tipo di destinazione.

Questo oggetto contiene le seguenti informazioni:

`topicMappingNameKey`

Il nome di questa mappatura degli argomenti. Sostituisci *`topicMappingNameKey`* con un nome che ti aiuti a identificare questa mappatura degli argomenti.

Questo oggetto contiene le seguenti informazioni:

`topic`

L'argomento o il filtro degli argomenti da collegare tra il broker di origine e quello di destinazione.

È possibile utilizzare i caratteri jolly degli argomenti + e # MQTT per inoltrare messaggi su tutti gli argomenti che corrispondono a un filtro per argomento. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide. AWS IoT Core

> **Note**
>
> Per utilizzare i caratteri jolly degli argomenti MQTT con il broker dei `Pubsub` sorgenti, è necessario utilizzare la versione 2.6.0 o successiva del componente Greengrass nucleus.

`targetTopicPrefix`

Il prefisso da aggiungere all'argomento di destinazione quando questo componente inoltra il messaggio.

`source`

Il broker di messaggi di origine. Seleziona una delle opzioni seguenti:

- `LocalMqtt`— Il broker MQTT locale con cui comunicano i dispositivi client.

- `Pubsub`— Il broker di messaggi di pubblicazione/sottoscrizione locale Greengrass.
- `IotCore`— Il AWS IoT Core broker di messaggi MQTT.

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

`source`e `target` deve essere diverso.

`target`

Il broker di messaggi di destinazione. Seleziona una delle opzioni seguenti:

- `LocalMqtt`— Il broker MQTT locale con cui comunicano i dispositivi client.
- `Pubsub`— Il broker di messaggi di pubblicazione/sottoscrizione locale Greengrass.
- `IotCore`— Il AWS IoT Core broker di messaggi MQTT.

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

`source`e `target` deve essere diverso.

`brokerUri`

(Facoltativo) L'URI del broker MQTT locale. È necessario specificare questo parametro se si configura il broker MQTT per utilizzare una porta diversa dalla porta predefinita 8883.

Utilizzate il seguente formato e sostituite la *porta* con la quale opera il broker MQTT:. `ssl://localhost:`*port*

Impostazione predefinita: `ssl://localhost:8883`

Example Esempio: fusione e aggiornamento della configurazione

Il seguente esempio di aggiornamento della configurazione specifica quanto segue:

- Inoltra i messaggi dai dispositivi client agli AWS IoT Core argomenti che corrispondono al filtro degli `clients/+/hello/world` argomenti.
- Inoltra i messaggi dai dispositivi client alla pubblicazione o sottoscrizione locale su argomenti che corrispondono al filtro degli argomenti e aggiungi il `events/input/` prefisso all'`clients/+/detections`argomento di destinazione. L'argomento di destinazione risultante corrisponde al filtro degli argomenti. `events/input/clients/+/detections`
- Inoltra i messaggi dai dispositivi client AWS IoT Core agli argomenti che corrispondono al filtro degli argomenti e aggiungi il `$aws/rules/StatusUpdateRule/` prefisso all'`clients/+/status`argomento di destinazione. Questo esempio inoltra questi messaggi direttamente a una AWS IoTregola denominata per `StatusUpdateRule` ridurre i costi utilizzando Basic Ingest.

```
{
  "mqttTopicMapping": {
    "ClientDeviceHelloWorld": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "ClientDeviceEvents": {
      "topic": "clients/+/detections",
      "targetTopicPrefix": "events/input/",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ClientDeviceCloudStatusUpdate": {
      "topic": "clients/+/status",
      "targetTopicPrefix": "$aws/rules/StatusUpdateRule/",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
```

```
  }
}
```

2.1.x

`mqttTopicMapping`

Le mappature degli argomenti che vuoi collegare. Questo componente sottoscrive i messaggi sull'argomento di origine e pubblica i messaggi che riceve nell'argomento di destinazione. Ogni mappatura degli argomenti definisce l'argomento, il tipo di origine e il tipo di destinazione.

Questo oggetto contiene le seguenti informazioni:

`topicMappingNameKey`

Il nome di questa mappatura degli argomenti. Sostituisci *`topicMappingNameKey`* con un nome che ti aiuti a identificare questa mappatura degli argomenti.

Questo oggetto contiene le seguenti informazioni:

`topic`

L'argomento o il filtro degli argomenti da collegare tra il broker di origine e quello di destinazione.

Se si specifica il broker `LocalMqtt` o il broker di `IotCore` origine, è possibile utilizzare i caratteri jolly degli argomenti + e # MQTT per inoltrare messaggi su tutti gli argomenti che corrispondono a un filtro di argomento. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide. AWS IoT Core

`source`

Il broker di messaggi di origine. Seleziona una delle opzioni seguenti:

- `LocalMqtt`— Il broker MQTT locale con cui comunicano i dispositivi client.
- `Pubsub`— Il broker di messaggi di pubblicazione/sottoscrizione locale Greengrass.
- `IotCore`— Il AWS IoT Core broker di messaggi MQTT.

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi

client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

`source`e `target` deve essere diverso.

`target`

Il broker di messaggi di destinazione. Seleziona una delle opzioni seguenti:

- `LocalMqtt`— Il broker MQTT locale con cui comunicano i dispositivi client.
- `Pubsub`— Il broker di messaggi di pubblicazione/sottoscrizione locale Greengrass.
- `IotCore`— Il AWS IoT Core broker di messaggi MQTT.

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

`source`e `target` deve essere diverso.

`brokerUri`

(Facoltativo) L'URI del broker MQTT locale. È necessario specificare questo parametro se si configura il broker MQTT per utilizzare una porta diversa dalla porta predefinita 8883. Utilizzate il seguente formato e sostituite la *`porta`* con la quale opera il broker MQTT:. `ssl://localhost:`*`port`*

Impostazione predefinita: `ssl://localhost:8883`

Example Esempio: fusione e aggiornamento della configurazione

L'esempio seguente di aggiornamento della configurazione specifica di inoltrare i messaggi dai dispositivi client agli argomenti AWS IoT Core on and. `clients/MyClientDevice1/hello/world` `clients/MyClientDevice2/hello/world`

```
{
  "mqttTopicMapping": {
    "ClientDevice1HelloWorld": {
      "topic": "clients/MyClientDevice1/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "ClientDevice2HelloWorld": {
      "topic": "clients/MyClientDevice2/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

2.0.x

`mqttTopicMapping`

Le mappature degli argomenti che si desidera collegare. Questo componente sottoscrive i messaggi sull'argomento di origine e pubblica i messaggi che riceve nell'argomento di destinazione. Ogni mappatura degli argomenti definisce l'argomento, il tipo di origine e il tipo di destinazione.

Questo oggetto contiene le seguenti informazioni:

`topicMappingNameKey`

Il nome di questa mappatura degli argomenti. Sostituisci *`topicMappingNameKey`* con un nome che ti aiuti a identificare questa mappatura degli argomenti.

Questo oggetto contiene le seguenti informazioni:

`topic`

L'argomento o il filtro degli argomenti da collegare tra il broker di origine e quello di destinazione.

Se si specifica il broker `LocalMqtt` o il broker di `IotCore` origine, è possibile utilizzare i caratteri jolly degli argomenti + e # MQTT per inoltrare messaggi su tutti gli argomenti che corrispondono a un filtro di argomento. Per ulteriori informazioni, consultate gli argomenti relativi a MQTT nella Developer Guide. AWS IoT Core

`source`

Il broker di messaggi di origine. Seleziona una delle opzioni seguenti:

- `LocalMqtt`— Il broker MQTT locale con cui comunicano i dispositivi client.
- `Pubsub`— Il broker di messaggi di pubblicazione/sottoscrizione locale Greengrass.
- `IotCore`— Il AWS IoT Core broker di messaggi MQTT.

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

`source`e `target` deve essere diverso.

`target`

Il broker di messaggi di destinazione. Seleziona una delle opzioni seguenti:

- `LocalMqtt`— Il broker MQTT locale con cui comunicano i dispositivi client.
- `Pubsub`— Il broker di messaggi di pubblicazione/sottoscrizione locale Greengrass.
- `IotCore`— Il AWS IoT Core broker di messaggi MQTT.

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

`source`e `target` deve essere diverso.

Example Esempio: fusione e aggiornamento della configurazione

L'esempio seguente di aggiornamento della configurazione specifica di inoltrare i messaggi dai dispositivi client agli argomenti AWS IoT Core on and. `clients/MyClientDevice1/hello/world` `clients/MyClientDevice2/hello/world`

```
{
  "mqttTopicMapping": {
    "ClientDevice1HelloWorld": {
      "topic": "clients/MyClientDevice1/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    },
    "ClientDevice2HelloWorld": {
      "topic": "clients/MyClientDevice2/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

Linux

```
sudo tail -f /greengrass/v2/logs/greengrass.log
```

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
```

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
2.3.1	Correzioni di bug e miglioramenti Risolve un problema a causa del quale il client MQTT locale entrava in un ciclo di disconnessione.
2.3.0	Nuove funzionalità Aggiunge il supporto MQTT5 per il collegamento tra sorgenti MQTT locali. AWS IoT Core
2.2.6	Nuove funzionalità Aggiunge una nuova opzione di configurazione. `startupTimeoutSeconds`
2.2.5	Versione aggiornata per la versione 2.4.0 di autenticazione dei dispositivi client.
2.2.4	Versione aggiornata per la versione 2.3.0 di autenticazione del dispositivo client Greengrass.
2.2.3	Questa versione contiene correzioni di bug e miglioramenti.

Versione	Modifiche
2.2.2	Correzioni di bug e miglioramenti • Modifiche alla registrazione.
2.2.1	Correzioni di bug e miglioramenti Risolve i problemi che possono causare la mancata sottoscrizione degli argomenti MQTT da parte del bridge MQTT.
2.2.0	Nuove funzionalità • Aggiunge il supporto per i caratteri jolly degli argomenti MQTT (#and+) quando si specifica local publish/subscribe come broker di messaggi di origine. Questa funzionalità richiede la versione 2.6.0 o successiva del componente Greengrass nucleus. • Aggiunge l'`targetTopicPrefix` opzione, che è possibile specificare per configurare il bridge MQTT per aggiungere un prefisso all'argomento di destinazione quando inoltra un messaggio.
2.1.1	Correzioni di bug e miglioramenti • Risolve i problemi relativi al modo in cui questo componente gestisce gli aggiornamenti di ripristino della configurazione. • Riduce la frequenza delle disconnessioni del client MQTT quando i certificati ruotano.
2.1.0	Nuove funzionalità • Aggiunge il `brokerUri` parametro, che consente di utilizzare una porta broker MQTT non predefinita.
2.0.1	Questa versione include correzioni di bug e miglioramenti.
2.0.0	Versione iniziale.

Broker MQTT 3.1.1 (Moquette)

Il componente broker Moquette MQTT (`aws.greengrass.clientdevices.mqtt.Moquette`) gestisce i messaggi MQTT tra i dispositivi client e un dispositivo core Greengrass. Questo componente fornisce una versione modificata del broker Moquette MQTT. Implementa questo broker MQTT per gestire un broker MQTT leggero. Per ulteriori informazioni su come scegliere un broker MQTT, consulta. Scegli un broker MQTT

Questo broker implementa il protocollo MQTT 3.1.1. Include il supporto per i messaggi conservati di QoS 0, QoS 1, QoS 2, i messaggi di ultima volontà e le sessioni persistenti.

> **Note**
>
> I dispositivi client sono dispositivi IoT locali che si connettono a un dispositivo core Greengrass per inviare messaggi MQTT e dati da elaborare. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.3.x
- 2.2.x
- 2.1.x

- 2,0x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Il dispositivo principale deve essere in grado di accettare connessioni sulla porta in cui opera il broker MQTT. Per impostazione predefinita, questo componente esegue il broker MQTT sulla porta 8883. È possibile specificare una porta diversa quando si configura questo componente.

  Se si specifica una porta diversa e si utilizza il componente bridge MQTT per inoltrare messaggi MQTT ad altri broker, è necessario utilizzare MQTT bridge v2.1.0 o versione successiva. Configuralo per utilizzare la porta su cui opera il broker MQTT.

  Se si specifica una porta diversa e si utilizza il componente IP detector per gestire gli endpoint del broker MQTT, è necessario utilizzare IP detector v2.1.0 o versione successiva. Configuralo per segnalare la porta su cui opera il broker MQTT.
- Il componente broker Moquette MQTT è supportato per l'esecuzione in un VPC.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle sue dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.3.2 – 2.3.6

La tabella seguente elenca le dipendenze per le versioni da 2.3.2 a 2.3.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.2.0 <2.5.0	Rigidi

2.3.0 and 2.3.1

La tabella seguente elenca le dipendenze per le versioni 2.3.0 e 2.3.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.2.0 <2.4.0	Rigidi

2.2.0

La tabella seguente elenca le dipendenze per la versione 2.2.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.2.0 <2.3.0	Rigidi

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.0.0 <2.2.0	Rigidi

2.0.0 - 2.0.2

La tabella seguente elenca le dipendenze per le versioni da 2.0.0 a 2.0.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.0.0 <2.1.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`moquette`

(Facoltativo) La configurazione del broker Moquette MQTT da utilizzare. È possibile configurare un sottoinsieme di opzioni di configurazione Moquette in questo componente. Per ulteriori informazioni, consultate i commenti in linea nel file di configurazione di Moquette.

Questo oggetto contiene le seguenti informazioni:

`ssl_port`

(Facoltativo) La porta in cui opera il broker MQTT.

> **Note**
>
> Se si specifica una porta diversa e si utilizza il componente bridge MQTT per inoltrare messaggi MQTT ad altri broker, è necessario utilizzare MQTT bridge v2.1.0 o versione successiva. Configuralo per utilizzare la porta su cui opera il broker MQTT.
> Se si specifica una porta diversa e si utilizza il componente IP detector per gestire gli endpoint del broker MQTT, è necessario utilizzare IP detector v2.1.0 o versione successiva. Configuralo per segnalare la porta su cui opera il broker MQTT.

Impostazione predefinita: `8883`

`host`

(Facoltativo) L'interfaccia a cui si collega il broker MQTT. Ad esempio, è possibile modificare questo parametro in modo che il broker MQTT si colleghi solo a una rete locale specifica.

Impostazione predefinita: `0.0.0.0` (si collega a tutte le interfacce di rete)

startupTimeoutSeconds

(Facoltativo) Il tempo massimo, in secondi, di avvio del componente. Lo stato del componente cambia `BROKEN` se supera questo timeout.

Impostazione predefinita: `120`

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica di utilizzare il broker MQTT sulla porta 443.

```
{
  "moquette": {
    "ssl_port": "443"
  }
}
```

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
2.3.6	Correzioni di bug e miglioramenti • Correzioni di bug generali e miglioramenti.
2.3.5	Correzioni di bug e miglioramenti • Moquette aggiornata alla versione 0.17.
2.3.4	Correzioni di bug e miglioramenti • Risolve un problema per cui i client potrebbero riscontrare errori di sessione non validi durante l'invio o la ricezione di messaggi, a causa

Versione	Modifiche
	di ID client duplicati. Questo problema ha causato la chiusura della sessione del client.
2.3.3	Nuove funzionalità Aggiunge una nuova opzione `startupTimeoutSeconds` di configurazione.
2.3.2	Versione aggiornata per la versione 2.4.0 di autenticazione dei dispositivi client.
2.3.1	Correzioni di bug e miglioramenti • Risolve una condizione di gara in cui i client potevano essere disconnessi dopo aver tentato di riconnettersi a causa di una sessione non valida.
2.3.0	Aggiunge il supporto per le catene di certificati.
2.2.0	Versione aggiornata per la versione 2.2.0 di autenticazione dei dispositivi client.
2.1.0	Correzioni di bug e miglioramenti • Aggiorna questo componente per utilizzare la versione 0.16 di Moquette, che migliora le prestazioni e include diversi altri miglioramenti. • Risolve un problema per cui il certificato del server MQTT locale ruota più spesso del previsto in determinati scenari. Per applicare questa correzione, è necessario utilizzare anche la versione 2.1.0 o successiva del componente di autenticazione del dispositivo client.
2.0.2	Correzioni di bug e miglioramenti • Aumenta la dimensione massima dei messaggi MQTT da 8.092 byte a 128 kilobyte. Il limite effettivo di payload dei messaggi MQTT è leggermente inferiore, poiché il limite di dimensione dei messaggi include le intestazioni dei messaggi. • Aggiunge il supporto per i valori interi nel parametro. `ssl_port`

Versione	Modifiche
2.0.1	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.0	Versione iniziale.

Broker MQTT 5 (EMQX)

Il componente broker EMQX MQTT (`aws.greengrass.clientdevices.mqtt.EMQX`) gestisce i messaggi MQTT tra i dispositivi client e un dispositivo core Greengrass. Questo componente fornisce una versione modificata del broker EMQX MQTT 5.0. Implementate questo broker MQTT per utilizzare le funzionalità MQTT 5 nella comunicazione tra i dispositivi client e un dispositivo principale. Per ulteriori informazioni su come scegliere un broker MQTT, vedere. Scegli un broker MQTT

Questo broker implementa il protocollo MQTT 5.0. Include il supporto per gli intervalli di scadenza delle sessioni e dei messaggi, le proprietà degli utenti, gli abbonamenti condivisi, gli alias degli argomenti e altro ancora. MQTT 5 è retrocompatibile con MQTT 3.1.1, quindi se si utilizza il broker Moquette MQTT 3.1.1, è possibile sostituirlo con il broker EMQX MQTT 5 e i dispositivi client possono continuare a connettersi e funzionare come al solito.

> **Note**
>
> I dispositivi client sono dispositivi IoT locali che si connettono a un dispositivo core Greengrass per inviare messaggi MQTT e dati da elaborare. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

Argomenti

- Versioni
- Type
- Sistema operativo
- Requisiti
- Dipendenze
- Configurazione
- File di registro locale

- Licenze
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 2.0.x
- 1.2.x
- 1.1.x
- 1.0.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Il dispositivo principale deve essere in grado di accettare connessioni sulla porta in cui opera il broker MQTT. Per impostazione predefinita, questo componente esegue il broker MQTT sulla porta 8883. È possibile specificare una porta diversa quando si configura questo componente.

  Se si specifica una porta diversa e si utilizza il componente bridge MQTT per inoltrare messaggi MQTT ad altri broker, è necessario utilizzare MQTT bridge v2.1.0 o versione successiva. Configuralo per utilizzare la porta su cui opera il broker MQTT.

Se si specifica una porta diversa e si utilizza il componente IP detector per gestire gli endpoint del broker MQTT, è necessario utilizzare IP detector v2.1.0 o versione successiva. Configuralo per segnalare la porta su cui opera il broker MQTT.

- Sui dispositivi principali Linux, Docker è installato e configurato sul dispositivo principale:
  - Docker Engine 1.9.1 o versione successiva installato sul dispositivo principale Greengrass. La versione 20.10 è l'ultima versione verificata per funzionare con il software Core. AWS IoT Greengrass È necessario installare Docker direttamente sul dispositivo principale prima di distribuire componenti che eseguono contenitori Docker.
  - Il daemon Docker è stato avviato e funzionante sul dispositivo principale prima di distribuire questo componente.
  - L'utente di sistema che esegue questo componente deve disporre delle autorizzazioni di root o amministratore. In alternativa, è possibile eseguire questo componente come utente di sistema nel `docker` gruppo e configurare l'`requiresPrivileges`opzione di questo componente per `false` eseguire il broker MQTT EQMX senza privilegi.
- Il componente broker EMQX MQTT è supportato per l'esecuzione in un VPC.
- Il componente broker EMQX MQTT non è supportato sulla piattaforma. `armv7`

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.0.0

La tabella seguente elenca le dipendenze per la versione 2.0.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.2.0 <2.5.0	Rigidi

1.2.2 – 1.2.3

La tabella seguente elenca le dipendenze per le versioni da 1.2.2 a 1.2.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.2.0 <2.5.0	Rigidi

1.2.0 and 1.2.1

La tabella seguente elenca le dipendenze per le versioni 1.2.0 e 1.2.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.2.0 <2.4.0	Rigidi

1.0.0 and 1.1.0

La tabella seguente elenca le dipendenze per le versioni 1.0.0 e 1.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Autenticazione del dispositivo client	>=2.2.0 <2.3.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

2.0.0

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

> **Important**
>
> Se si utilizza la versione 2 del componente del broker MQTT 5 (EMQX), è necessario aggiornare il file di configurazione. I file di configurazione della versione 1 non funzionano con la versione 2.

emqxConfig

(Facoltativo) La configurazione del broker EMQX MQTT da utilizzare. È possibile impostare le opzioni di configurazione EMQX in questo componente.

Quando si utilizza il broker EMQX, Greengrass utilizza una configurazione predefinita. Questa configurazione viene utilizzata a meno che non venga modificata utilizzando questo campo.

La modifica delle seguenti impostazioni di configurazione causa il riavvio del componente del broker EMQX. Le altre modifiche alla configurazione si applicano senza riavviare il componente.

- `emqxConfig/cluster`
- `emqxConfig/node`
- `emqxConfig/rpc`

> **Note**
>
> `aws.greengrass.clientdevices.mqtt.EMQX`consente di configurare opzioni sensibili alla sicurezza. Questi includono impostazioni TLS, autenticazione e provider di autorizzazione. Abbiamo consigliato la configurazione predefinita che utilizza l'autenticazione TLS reciproca e il provider di autenticazione del dispositivo client Greengrass.

Example Esempio: configurazione predefinita

L'esempio seguente mostra i valori predefiniti impostati per il broker MQTT 5 (EMQX). È possibile sovrascrivere queste impostazioni utilizzando l'impostazione di configurazione. `emqxConfig`

```
{
  "authorization": {
```

```
    "no_match": "deny",
    "sources": []
  },
  "node": {
    "cookie": "<placeholder>"
  },
  "listeners": {
     "ssl": {
       "default": {
         "ssl_options": {
           "keyfile": "{work:path}\\data\\key.pem",
           "certfile": "{work:path}\\data\\cert.pem",
           "cacertfile": null,
           "verify": "verify_peer",
           "versions": ["tlsv1.3", "tlsv1.2"],
           "fail_if_no_peer_cert": true
         }
       }
     },
     "tcp": {
       "default": {
         "enabled": false
       }
     },
     "ws": {
       "default": {
         "enabled": false
       }
     },
     "wss": {
       "default": {
         "enabled": false
       }
     }
  },
  "plugins": {
    "states": [{"name_vsn": "gg-1.0.0", "enable": true}],
    "install_dir": "plugins"
  }
}
```

AuthMode

(Facoltativo) Imposta il provider di autorizzazione per il broker. Può essere uno dei seguenti valori:

- `enabled`— (Impostazione predefinita) Utilizza il provider di autenticazione e autorizzazione Greengrass.
- `bypass_on_failure`— Utilizzare il provider di autenticazione Greengrass, quindi utilizzare tutti i provider di autenticazione rimanenti nella catena di provider EMQX se Greengrass nega l'autenticazione o l'autorizzazione.
- `bypass`— Il provider Greengrass è disabilitato. L'autenticazione e l'autorizzazione sono gestite dalla catena di fornitori EMQX.

`requiresPrivilege`

(Facoltativo) Sui dispositivi principali Linux, è possibile specificare di eseguire il broker EMQX MQTT senza privilegi di root o amministratore. Se si imposta questa opzione su`false`, l'utente di sistema che esegue questo componente deve essere un membro del gruppo. `docker`

Impostazione predefinita: `true`

`startupTimeoutSeconds`

(Facoltativo) Il tempo massimo in secondi per l'avvio del broker EMQX MQTT. Lo stato del componente cambia `BROKEN` se supera questo timeout.

Impostazione predefinita: `90`

`ipcTimeoutSeconds`

(Facoltativo) Il tempo massimo, in secondi, impiegato dal componente per attendere che il nucleo Greengrass risponda alle richieste di comunicazione tra processi (IPC). Aumentate questo numero se questo componente segnala errori di timeout quando verifica se un dispositivo client è autorizzato.

Impostazione predefinita: `5`

`crtLogLevel`

(Facoltativo) Il livello di registro per la libreria AWS Common Runtime (CRT).

Il valore predefinito è il livello di log del broker EMQX MQTT (in). `log.level` `emqx`

`restartIdentifier`

(Facoltativo) Configurare questa opzione per riavviare il broker EMQX MQTT. Quando questo valore di configurazione cambia, questo componente riavvia il broker MQTT. È possibile utilizzare questa opzione per forzare la disconnessione dei dispositivi client.

`dockerOptions`

(Facoltativo) Configura questa opzione solo sui sistemi operativi Linux per aggiungere parametri alla riga di comando Docker. Ad esempio, per mappare porte aggiuntive, usa il parametro `-p` Docker:

```
"-p 1883:1883"
```

Example Esempio: aggiornamento di un file di configurazione v1.x alla v2.x

L'esempio seguente mostra le modifiche necessarie per aggiornare un file di configurazione v1.x alla versione 2.x.

Il file di configurazione della versione 1.x:

```
{
    "emqx": {
        "listener.ssl.external": "443",
        "listener.ssl.external.max_connections": "1024000",
        "listener.ssl.external.max_conn_rate": "500",
        "listener.ssl.external.rate_limit": "50KB,5s",
        "listener.ssl.external.handshake_timeout": "15s",
        "log.level": "warning"
    },
    "mergeConfigurationFiles": {
        "etc/plugins/aws_greengrass_emqx_auth.conf": "auth_mode=enabled\n
 use_greengrass_managed_certificates=true\n"
    }
}
```

Il file di configurazione equivalente per la v2:

```
{
    "emqxConfig": {
        "listeners": {
            "ssl": {
```

```
                "default": {
                   "bind": "8883",
                   "max_connections": "1024000",
                   "max_conn_rate": "500",
                   "handshake_timeout": "15s"
                }
            }
        },
        "log": {
            "console": {
              "enable": true,
              "level": "warning"
            }
        }
    },
    "authMode": "enabled"
}
```

Non esiste un equivalente alla voce di `listener.ssl.external.rate_limit` configurazione. L'opzione `use_greengrass_managed_certificates` di configurazione è stata rimossa.

Example Esempio: imposta una nuova porta per il broker

L'esempio seguente modifica la porta su cui opera il broker MQTT dalla porta predefinita 8883 alla porta 1234. Se utilizzi Linux, includi il `dockerOptions` campo.

```
{
  "emqxConfig": {
    "listeners": {
      "ssl": {
        "default": {
          "bind": 1234
        }
      }
    }
  },
  "dockerOptions": "-p 1234:1234"
}
```

Example Esempio: modifica il livello di log del broker MQTT

L'esempio seguente modifica il livello di log del broker MQTT in. `debug` È possibile scegliere tra i seguenti livelli di registro:

- `debug`
- `info`
- `notice`
- `warning`
- `error`
- `critical`
- `alert`
- `emergency`

Il livello di registro predefinito è`warning`.

```
{
  "emqxConfig": {
    "log": {
      "console": {
         "level": "debug"
      }
    }
  }
}
```

Example Esempio: abilitare il pannello di controllo EMQX

L'esempio seguente abilita la dashboard EMQX in modo da poter monitorare e gestire il broker. Se utilizzi Linux, includi il `dockerOptions` campo.

```
{
  "emqxConfig": {
    "dashboard": {
      "listeners": {
        "http": {
          "bind": 18083
        }
      }
    }
  },
  "dockerOptions": "-p 18083:18083"
}
```

1.0.0 - 1.2.2

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`emqx`

(Facoltativo) La configurazione del broker EMQX MQTT da utilizzare. È possibile configurare un sottoinsieme di opzioni di configurazione EMQX in questo componente.

Questo oggetto contiene le seguenti informazioni:

`listener.ssl.external`

(Facoltativo) La porta in cui opera il broker MQTT.

> **Note**
>
> Se si specifica una porta diversa e si utilizza il componente bridge MQTT per inoltrare messaggi MQTT ad altri broker, è necessario utilizzare MQTT bridge v2.1.0 o versione successiva. Configuralo per utilizzare la porta su cui opera il broker MQTT.
> Se si specifica una porta diversa e si utilizza il componente IP detector per gestire gli endpoint del broker MQTT, è necessario utilizzare IP detector v2.1.0 o versione successiva. Configuralo per segnalare la porta su cui opera il broker MQTT.

Impostazione predefinita: `8883`

`listener.ssl.external.max_connections`

(Facoltativo) Il numero massimo di connessioni simultanee supportate dal broker MQTT.

Impostazione predefinita: `1024000`

`listener.ssl.external.max_conn_rate`

(Facoltativo) Il numero massimo di nuove connessioni al secondo che il broker MQTT può ricevere.

Impostazione predefinita: `500`

`listener.ssl.external.rate_limit`

(Facoltativo) Il limite di larghezza di banda per tutte le connessioni al broker MQTT. Specificate la larghezza di banda e la durata di tale larghezza di banda separate da una virgola (`,`) nel seguente formato:. `bandwidth,duration` Ad esempio, è possibile specificare di `50KB,5s` limitare il broker MQTT a 50 kilobyte (KB) di dati ogni 5 secondi.

`listener.ssl.external.handshake_timeout`

(Facoltativo) La quantità di tempo che il broker MQTT attende per completare l'autenticazione di una nuova connessione.

Impostazione predefinita: `15s`

`mqtt.max_packet_size`

(Facoltativo) La dimensione massima di un messaggio MQTT.

Impostazione predefinita: `268435455` (256 MB meno 1)

`log.level`

(Facoltativo) Il livello di log per il broker MQTT. Seleziona una delle opzioni seguenti:

- `debug`
- `info`
- `notice`
- `warning`
- `error`
- `critical`
- `alert`
- `emergency`

Il livello di registro predefinito è`warning`.

`requiresPrivilege`

(Facoltativo) Sui dispositivi principali Linux, è possibile specificare di eseguire il broker EMQX MQTT senza privilegi di root o amministratore. Se si imposta questa opzione su`false`, l'utente di sistema che esegue questo componente deve essere un membro del gruppo. `docker`

Impostazione predefinita: `true`

`startupTimeoutSeconds`

(Facoltativo) Il tempo massimo in secondi per l'avvio del broker EMQX MQTT. Lo stato del componente cambia `BROKEN` se supera questo timeout.

Impostazione predefinita: `90`

`ipcTimeoutSeconds`

(Facoltativo) Il tempo massimo, in secondi, impiegato dal componente per attendere che il nucleo Greengrass risponda alle richieste di comunicazione tra processi (IPC). Aumentate questo numero se questo componente segnala errori di timeout quando verifica se un dispositivo client è autorizzato.

Impostazione predefinita: `5`

`crtLogLevel`

(Facoltativo) Il livello di registro per la libreria AWS Common Runtime (CRT).

Il valore predefinito è il livello di log del broker EMQX MQTT (in). `log.level` `emqx`

`restartIdentifier`

(Facoltativo) Configurare questa opzione per riavviare il broker EMQX MQTT. Quando questo valore di configurazione cambia, questo componente riavvia il broker MQTT. È possibile utilizzare questa opzione per forzare la disconnessione dei dispositivi client.

`dockerOptions`

(Facoltativo) Configura questa opzione solo sui sistemi operativi Linux per aggiungere parametri alla riga di comando Docker. Ad esempio, per mappare porte aggiuntive, usa il parametro `-p` Docker:

```
"-p 1883:1883"
```

`mergeConfigurationFiles`

(Facoltativo) Configurate questa opzione per aggiungere o sostituire i valori predefiniti nei file di configurazione EMQX specificati. Per informazioni sui file di configurazione e sui relativi formati, vedere Configurazione nella documentazione di EMQX 4.0. I valori specificati vengono aggiunti al file di configurazione.

L'esempio seguente aggiorna il `etc/emqx.conf` file.

```
"mergeConfigurationFiles": {
    "etc/emqx.conf": "broker.sys_interval=30s\nbroker.sys_heartbeat=10s"
},
```

Oltre ai file di configurazione supportati da EMQX, Greengrass supporta un file che configura il plugin di autenticazione Greengrass per EMQX chiamato. `etc/plugins/aws_greengrass_emqx_auth.conf` Sono supportate due opzioni e. `auth_mode` `use_greengrass_managed_certificates` Per utilizzare un altro provider di autenticazione, imposta l'`auth_mode`opzione su una delle seguenti:

- `enabled`— (Impostazione predefinita) Utilizza il provider di autenticazione e autorizzazione Greengrass.
- `bypass_on_failure`— Utilizzare il provider di autenticazione Greengrass, quindi utilizzare tutti i provider di autenticazione rimanenti nella catena di provider EMQX se Greengrass nega l'autenticazione o l'autorizzazione.
- `bypass`— Il provider Greengrass è disabilitato. L'autenticazione e l'autorizzazione vengono quindi gestite dalla catena di fornitori EMQX.

In caso `use_greengrass_managed_certificates` affermativo`true`, questa opzione indica che Greengrass gestisce i certificati TLS del broker. Se`false`, indica che i certificati vengono forniti tramite un'altra fonte.

L'esempio seguente aggiorna le impostazioni predefinite nel `etc/plugins/aws_greengrass_emqx_auth.conf` file di configurazione.

```
"mergeConfigurationFiles": {
    "etc/plugins/aws_greengrass_emqx_auth.conf": "auth_mode=enabled\n
 use_greengrass_managed_certificates=true\n"
  },
```

> **Note**
>
> `aws.greengrass.clientdevices.mqtt.EMQX`consente di configurare opzioni sensibili alla sicurezza. Questi includono impostazioni TLS, autenticazione e provider di autorizzazione. La configurazione consigliata è la configurazione predefinita che utilizza l'autenticazione TLS reciproca e il provider Greengrass Client Device Auth.

`replaceConfigurationFiles`

(Facoltativo) Configurate questa opzione per sostituire i file di configurazione EMQX specificati. I valori specificati sostituiscono l'intero file di configurazione esistente. Non è possibile specificare il `etc/emqx.conf` file in questa sezione. È necessario utilizzare `mergeConfigurationFile` per modificare`etc/emqx.conf`.

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica di utilizzare il broker MQTT sulla porta 443.

```
{
  "emqx": {
    "listener.ssl.external": "443",
    "listener.ssl.external.max_connections": "1024000",
    "listener.ssl.external.max_conn_rate": "500",
    "listener.ssl.external.rate_limit": "50KB,5s",
    "listener.ssl.external.handshake_timeout": "15s",
    "log.level": "warning"
  },
  "requiresPrivilege": "true",
  "startupTimeoutSeconds": "90",
  "ipcTimeoutSeconds": "5"
}
```

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.clientdevices.mqtt.EMQX.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.clientdevices.mqtt.EMQX.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.greengrass.clientdevices.mqtt.EMQX.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.greengrass.clientdevices.mqtt.EMQX.log -
  Tail 10 -Wait
  ```

Licenze

Nei sistemi operativi Windows, questo software include codice distribuito in base alle Condizioni di licenza software Microsoft - Microsoft Visual Studio Community 2022. Scaricando questo software, l'utente accetta le condizioni di licenza del codice.

Questo componente è rilasciato in base al contratto di licenza del software Greengrass Core.

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

v2.x

Versione	Modifiche
2.0.0	Questa versione del broker MQTT 5 (EMQX) prevede parametri di configurazione diversi rispetto alla versione 1.x. Se si utilizza una configura zione non predefinita per la versione 1.x, è necessario aggiornare la configurazione del componente per la 2.x. Per ulteriori informazioni, consulta Configurazione. Nuove funzionalità • Aggiorna il broker MQTT a EMQX 5.1.1.

Versione	Modifiche
	• Abilita le modifiche alla configurazione del broker senza riavviare il componente. Aggiornamenti • Aggiunge un nuovo campo `emqxConfig` di configurazione che sostituisce i campi `emqxmergeConfigurationFiles` , e `replaceConfigurationFiles` di configurazione.

v1.x

Versione	Modifiche
1.2.3	Correzioni di bug e miglioramenti • Risolve un problema per cui i client non potevano interagire con EMQX dopo l'autenticazione precedente disconnettendo e riautenti cando il client.
1.2.2	Versione aggiornata per la versione 2.4.0 di autenticazione del dispositivo client.
1.2.1	Correzioni di bug e miglioramenti • Risolve un problema per cui il componente non si avvia in Windows se Visual C++ Redistributable non è già presente. • Aggiorna EMQX alla versione 4.4.14.
1.2.0	Aggiunge il supporto per le catene di certificati.
1.1.0	Nuove funzionalità • Aggiunge il supporto per le configurazioni EMQX, comprese le opzioni del broker e i plug-in. Correzioni di bug e miglioramenti • Aggiorna EMQX alla versione 4.4.9.
1.0.1	Risolve un problema durante l'handshake TLS che impediva la connessio ne di alcuni client MQTT.

Versione	Modifiche
1.0.0	Versione iniziale.

Emettitore di telemetria Nucleus

Il nucleus telemetry emitter component (`aws.greengrass.telemetry.NucleusEmitter`) raccoglie dati di telemetria sullo stato del sistema e li pubblica continuamente su un argomento locale e su un argomento MQTT. AWS IoT Core Questo componente consente di raccogliere dati di telemetria di sistema in tempo reale sui dispositivi principali Greengrass. Per informazioni sull'agente di telemetria Greengrass che pubblica i dati di telemetria di sistema su Amazon, consulta. EventBridge Raccogli dati di telemetria sanitaria del sistema dai dispositivi principali AWS IoT Greengrass

Per impostazione predefinita, il componente Nucleus Telemetry Emitter pubblica i dati di telemetria ogni 60 secondi nel seguente argomento di pubblicazione/sottoscrizione locale.

```
$local/greengrass/telemetry
```

Per impostazione predefinita, il componente dell'emettitore di telemetria nucleus non viene pubblicato su un argomento MQTT. AWS IoT Core È possibile configurare questo componente per la pubblicazione su un argomento AWS IoT Core MQTT al momento della distribuzione. L'uso di un argomento MQTT per pubblicare dati su Cloud AWS è soggetto a prezzi. AWS IoT Core

AWS IoT Greengrassfornisce diversi componenti della community per aiutarti ad analizzare e visualizzare i dati di telemetria localmente sul tuo dispositivo principale utilizzando InfluxDB e Grafana. Questi componenti utilizzano i dati di telemetria del componente emettitore del nucleo. Per ulteriori informazioni, consultate il README per il componente publisher InfluxDB.

Argomenti

- Versioni
- Type
- Sistema operativo
- Dipendenze
- Configurazione
- Dati di output

- Utilizzo
- File di registro locale
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 1.0.x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

1.0.8

La tabella seguente elenca le dipendenze per la versione 1.0.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.4.0 <2.13.0	Rigidi

1.0.7

La tabella seguente elenca le dipendenze per la versione 1.0.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.4.0 <2.12.0	Rigidi

1.0.6

La tabella seguente elenca le dipendenze per la versione 1.0.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.4.0 <2.11.0	Rigidi

1.0.5

La tabella seguente elenca le dipendenze per la versione 1.0.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.4.0 <2.10.0	Rigidi

1.0.4

La tabella seguente elenca le dipendenze per la versione 1.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.4.0 <2.9.0	Rigidi

1.0.3

La tabella seguente elenca le dipendenze per la versione 1.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.4.0 <2.8.0	Rigidi

1.0.2

La tabella seguente elenca le dipendenze per la versione 1.0.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.4.0 <2.7.0	Rigidi

1.0.1

La tabella seguente elenca le dipendenze per la versione 1.0.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.4.0 <2.6.0	Rigidi

1.0.0

La tabella seguente elenca le dipendenze per la versione 1.0.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.4.0 <2.5.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`pubSubPublish`

(Facoltativo) Definisce se pubblicare i dati di telemetria sull'argomento. `$local/greengrass/telemetry` I valori supportati sono `true` e `false`.

Impostazione predefinita: `true`

`mqttTopic`

(Facoltativo) L'argomento AWS IoT Core MQTT su cui questo componente pubblica i dati di telemetria.

Imposta questo valore sull'argomento AWS IoT Core MQTT su cui desideri pubblicare i dati di telemetria. Quando questo valore è vuoto, l'emettitore del nucleo non pubblica i dati di telemetria su. Cloud AWS

> **Note**
>
> L'uso di un argomento MQTT per pubblicare dati su è soggetto ai prezzi. Cloud AWSAWS IoT Core

Impostazione predefinita: `""`

`telemetryPublishIntervalMs`

(Facoltativo) La quantità di tempo (in millisecondi) tra cui il componente pubblica i dati di telemetria. Se imposti questo valore su un valore inferiore al valore minimo supportato, il componente utilizza invece il valore minimo.

> **Note**
>
> Intervalli di pubblicazione inferiori comportano un maggiore utilizzo della CPU sul dispositivo principale. Ti consigliamo di iniziare con l'intervallo di pubblicazione predefinito e di modificarlo in base all'utilizzo della CPU del dispositivo.

Minimo: `500`

Impostazione predefinita: `60000`

Example Esempio: fusione e aggiornamento della configurazione

L'esempio seguente mostra un esempio di aggiornamento di fusione della configurazione che consente la pubblicazione dei dati di telemetria ogni 5 secondi sull'argomento e sull'`$local/greengrass/telemetry`argomento MQTT. `greengrass/myTelemetry` AWS IoT Core

```
{
  "pubSubPublish": "true",
  "mqttTopic": "greengrass/myTelemetry",
  "telemetryPublishIntervalMs": 5000
}
```

Dati di output

Questo componente pubblica le metriche di telemetria come array JSON nell'argomento seguente.

Argomento locale: `$local/greengrass/telemetry`

Facoltativamente, puoi scegliere di pubblicare anche metriche di telemetria su un argomento MQTT. AWS IoT Core Per ulteriori informazioni sugli argomenti, consultate gli argomenti MQTT nella Developer Guide. AWS IoT Core

Example Dati di esempio

```
[
  {
    "A": "Average",
    "N": "CpuUsage",
    "NS": "SystemMetrics",
    "TS": 1627597331445,
```

```
    "U": "Percent",
    "V": 26.21981271562346
  },
  {
    "A": "Count",
    "N": "TotalNumberOfFDs",
    "NS": "SystemMetrics",
    "TS": 1627597331445,
    "U": "Count",
    "V": 7316
  },
  {
    "A": "Count",
    "N": "SystemMemUsage",
    "NS": "SystemMetrics",
    "TS": 1627597331445,
    "U": "Megabytes",
    "V": 10098
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsStarting",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsInstalled",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsStateless",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
```

```
    "A": "Count",
    "N": "NumberOfComponentsStopping",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsBroken",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsRunning",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 7
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsErrored",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsNew",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
    "U": "Count",
    "V": 0
  },
  {
    "A": "Count",
    "N": "NumberOfComponentsFinished",
    "NS": "GreengrassComponents",
    "TS": 1627597331446,
```

```
    "U": "Count",
    "V": 2
  }
]
```

L'array di output contiene un elenco di metriche con le seguenti proprietà:

`A`

Il tipo di aggregazione per la metrica.

Per la `CpuUsage` metrica, questa proprietà è impostata su `Average` perché il valore pubblicato della metrica è l'utilizzo medio della CPU dall'ultimo evento di pubblicazione.

Per tutte le altre metriche, l'emettitore del nucleo non aggrega il valore metrico e questa proprietà è impostata su. `Count`

`N`

Nome del parametro.

`NS`

Lo spazio dei nomi metrico.

`TS`

Il timestamp di quando i dati sono stati raccolti.

`U`

L'unità del valore metrico.

`V`

Valore del parametro .

L'emettitore del nucleo pubblica le seguenti metriche:

Nome	Descrizione	
System (Sistema)		
`SystemMemUsage`	La quantità di memoria attualmente utilizzata da tutte	

Nome	Descrizione
	le applicazioni sul dispositivo principale Greengrass, incluso il sistema operativo.
`CpuUsage`	La quantità di CPU attualmente utilizzata da tutte le applicazioni sul dispositivo principale Greengrass, incluso il sistema operativo.
`TotalNumberOfFDs`	Il numero di descrittori di file memorizzati dal sistema operativo del dispositivo principale Greengrass. Un descrittore di file identifica in modo univoco un file aperto.
Nucleo Greengrass	
`NumberOfComponentsRunning`	Il numero di componenti in esecuzione sul dispositivo principale Greengrass.
`NumberOfComponentsErrored`	Il numero di componenti che si trovano in stato di errore sul dispositivo principale Greengrass.
`NumberOfComponentsInstalled`	Il numero di componenti installati sul dispositivo principale Greengrass.
`NumberOfComponentsStarting`	Il numero di componenti che si avviano sul dispositivo principale Greengrass.

Nome	Descrizione
`NumberOfComponentsNew`	Il numero di componenti nuovi sul dispositivo principale Greengrass.
`NumberOfComponentsStopping`	Il numero di componenti che si arrestano sul dispositivo principale Greengrass.
`NumberOfComponentsFinished`	Il numero di componenti completati sul dispositivo principale Greengrass.
`NumberOfComponentsBroken`	Il numero di componenti danneggiati sul dispositivo principale Greengrass.
`NumberOfComponentsStateless`	Il numero di componenti stateless sul dispositivo principale Greengrass.

Utilizzo

Per utilizzare i dati di telemetria relativi allo stato del sistema, è possibile creare componenti personalizzati basati sugli argomenti in cui l'emettitore nucleare pubblica i dati di telemetria e reagire a tali dati in base alle esigenze. Poiché il componente nucleus emitter offre la possibilità di pubblicare dati di telemetria su un argomento locale, puoi abbonarti a quell'argomento e utilizzare i dati pubblicati per agire localmente sul tuo dispositivo principale. Il dispositivo principale può quindi reagire ai dati di telemetria anche quando ha una connettività limitata al cloud.

Ad esempio, è possibile configurare un componente che ascolta i dati di telemetria sull'`$local/greengrass/telemetry`argomento e inviare i dati al componente stream manager per trasmettere i dati a. Cloud AWS Per ulteriori informazioni sulla creazione di tale componente, consulta e. Pubblicare/sottoscrivere messaggi locali Crea componenti personalizzati che utilizzano stream manager

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
1.0.8	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
1.0.7	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
1.0.6	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.

Versione	Modifiche
1.0.5	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
1.0.4	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
1.0.3	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
1.0.2	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
1.0.1	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
1.0.0	Versione iniziale.

Fornitore PKCS #11

Il componente provider PKCS #11 (`aws.greengrass.crypto.Pkcs11Provider`) consente di configurare il software AWS IoT Greengrass Core per l'utilizzo di un modulo di sicurezza hardware (HSM) tramite l'interfaccia PKCS #11. Questo componente consente di archiviare in modo sicuro i file di certificati e chiavi private in modo che non vengano esposti o duplicati nel software. Per ulteriori informazioni, consulta Integrazione della sicurezza hardware.

Per effettuare il provisioning di un dispositivo core Greengrass che memorizza il certificato e la chiave privata in un HSM, è necessario specificare questo componente come plug-in di provisioning quando si installa il software Core. AWS IoT Greengrass Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning manuale delle risorse.

AWS IoT Greengrassfornisce questo componente come file JAR che potete scaricare e specificare come plug-in di provisioning durante l'installazione. È possibile scaricare la versione più recente del file JAR del componente al seguente URL: https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/aws.greengrass.crypto.Pkcs11Provider-latest.jar.

Argomenti

- Versioni
- Type
- Sistema operativo
- Requisiti
- Dipendenze

- Configurazione
- File di registro locale
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 2.0.x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato solo sui dispositivi principali di Linux.

Requisiti

Questo componente ha i seguenti requisiti:

- Un modulo di sicurezza hardware che supporta lo schema di firma PKCS #1 v1.5 e le chiavi RSA con chiave RSA-2048 (o superiore) o chiavi ECC.

> **Note**
>
> Per utilizzare un modulo di sicurezza hardware con chiavi ECC, è necessario utilizzare Greengrass nucleus v2.5.6 o versione successiva.
> Per utilizzare un modulo di sicurezza hardware e un gestore segreto, è necessario utilizzare un modulo di sicurezza hardware con chiavi RSA.

- Una libreria di provider PKCS #11 che il software AWS IoT Greengrass Core può caricare in fase di esecuzione (usando libdl) per richiamare le funzioni PKCS #11. La libreria del provider PKCS #11 deve implementare le seguenti operazioni API PKCS #11:
  - `C_Initialize`
  - `C_Finalize`
  - `C_GetSlotList`
  - `C_GetSlotInfo`
  - `C_GetTokenInfo`
  - `C_OpenSession`
  - `C_GetSessionInfo`
  - `C_CloseSession`
  - `C_Login`
  - `C_Logout`
  - `C_GetAttributeValue`
  - `C_FindObjectsInit`
  - `C_FindObjects`
  - `C_FindObjectsFinal`
  - `C_DecryptInit`
  - `C_Decrypt`
  - `C_DecryptUpdate`
  - `C_DecryptFinal`
  - `C_SignInit`
  - `C_Sign`
  - `C_SignUpdate`
  - `C_SignFinal`
  - `C_GetMechanismList`
  - `C_GetMechanismInfo`
  - `C_GetInfo`
  - `C_GetFunctionList`
- Il modulo hardware deve essere risolvibile mediante l'etichetta dello slot, come definito nella specifica PKCS#11.

- È necessario archiviare la chiave privata e il certificato nell'HSM nello stesso slot e utilizzare la stessa etichetta e lo stesso ID dell'oggetto, se l'HSM supporta gli ID degli oggetti.
- Il certificato e la chiave privata devono essere risolvibili mediante etichette di oggetti.
- La chiave privata deve avere le seguenti autorizzazioni:
  - `sign`
  - `decrypt`
- (Facoltativo) Per utilizzare il componente Secret Manager, è necessario utilizzare la versione 2.1.0 o successiva e la chiave privata deve disporre delle seguenti autorizzazioni:
  - `unwrap`
  - `wrap`
- (Facoltativo) Se si utilizza la libreria TPM2 e si esegue il core Greengrass come servizio, è necessario fornire una variabile di ambiente con la posizione dell'archivio PKCS #11. L'esempio seguente è un file di servizio systemd con la variabile di ambiente richiesta:

```
[Unit]
Description=Greengrass Core
After=network.target

[Service]
Type=simple
PIDFile=/var/run/greengrass.pid
Environment=TPM2_PKCS11_STORE=/path/to/store/directory
RemainAfterExit=no
Restart=on-failure
RestartSec=10
ExecStart=/bin/sh /greengrass/v2/alts/current/distro/bin/loader

[Install]
WantedBy=multi-user.target
```

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le

dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.0.7

La tabella seguente elenca le dipendenze per la versione 2.0.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.3 <2.13.0	Flessibili

2.0.6

La tabella seguente elenca le dipendenze per la versione 2.0.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.3 <2.12.0	Flessibili

2.0.5

La tabella seguente elenca le dipendenze per la versione 2.0.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.3 <2.11.0	Flessibili

2.0.4

La tabella seguente elenca le dipendenze per la versione 2.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.3 <2.10.0	Flessibili

2.0.3

La tabella seguente elenca le dipendenze per la versione 2.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.3 <2.9.0	Flessibili

2.0.2

La tabella seguente elenca le dipendenze per la versione 2.0.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.3 <2.8.0	Flessibili

2.0.1

La tabella seguente elenca le dipendenze per la versione 2.0.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.3 <2.7.0	Flessibili

2.0.0

La tabella seguente elenca le dipendenze per la versione 2.0.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.3 <2.6.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`name`

Un nome per la configurazione PKCS #11.

`library`

Il percorso assoluto del file alla libreria dell'implementazione PKCS #11 che il software AWS IoT Greengrass Core può caricare con libdl.

`slot`

L'ID dello slot che contiene la chiave privata e il certificato del dispositivo. Questo valore è diverso dall'indice o dall'etichetta dello slot.

`userPin`

Il PIN utente da utilizzare per accedere allo slot.

Example Esempio: fusione e aggiornamento della configurazione

```
{
  "name": "softhsm_pkcs11",
  "library": "/usr/lib/softhsm/libsofthsm2.so",
  "slot": 1,
  "userPin": "1234"
}
```

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.0.7	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.0.6	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.0.5	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.0.4	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.0.3	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.0.2	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.0.1	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.0.0	Versione iniziale.

Gestore segreto

Il componente secret manager (`aws.greengrass.SecretManager`) distribuisce i segreti dai dispositivi AWS Secrets Manager principali di Greengrass. Usa questo componente per utilizzare in modo sicuro le credenziali, come le password, nei componenti personalizzati dei tuoi dispositivi principali Greengrass. Per ulteriori informazioni su Secrets Manager, vedi Cos'èAWS Secrets Manager? nella Guida AWS Secrets Manager per l'utente.

Per accedere ai segreti di questo componente nei componenti Greengrass personalizzati, utilizzate l'GetSecretValueoperazione in. SDK per dispositivi AWS IoT Per ulteriori informazioni, consultare Usa il SDK per dispositivi AWS IoT per comunicare con il nucleo Greengrass, altri componenti e AWS IoT Core e Recupera valori segreti.

Questo componente crittografa i segreti sul dispositivo principale per proteggere le credenziali e le password fino al momento del loro utilizzo. Utilizza la chiave privata del dispositivo principale per crittografare e decrittografare i segreti.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x
- 2,0x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Il ruolo del dispositivo Greengrass deve consentire l'`secretsmanager:GetSecretValue`azione, come illustrato nel seguente esempio di policy IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "secretsmanager:GetSecretValue"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:secretsmanager:region:123456789012:secret:MySecret"
      ]
    }
  ]
}
```

> **Note**
>
> Se utilizzi una AWS Key Management Service chiave gestita dal cliente per crittografare i segreti, anche il ruolo del dispositivo deve consentire l'azione. `kms:Decrypt`

Per ulteriori informazioni sulle politiche IAM per Secrets Manager, consulta quanto segue nella Guida per l'AWS Secrets Managerutente:

- Autenticazione e controllo degli accessi per AWS Secrets Manager
- Azioni, risorse e chiavi di contesto che puoi utilizzare in una policy IAM o in una policy segreta per AWS Secrets Manager

- I componenti personalizzati devono definire una politica di autorizzazione che `aws.greengrass#GetSecretValue` consenta di ottenere i segreti archiviati con questo componente. In questa politica di autorizzazione, puoi limitare l'accesso dei componenti a segreti specifici. Per ulteriori informazioni, consulta l'autorizzazione IPC del gestore segreto.
- (Facoltativo) Se si archivia la chiave privata e il certificato del dispositivo principale in un modulo di sicurezza hardware (HSM), l'HSM deve supportare le chiavi RSA, la chiave privata deve disporre dell'`unwrap`autorizzazione e la chiave pubblica deve disporre dell'autorizzazione. `wrap`

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`secretsmanager.` *`region`*`.amazonaws.com`	443	Sì	Scarica i segreti sul dispositivo principale.

Dipendenze

Quando distribuisci un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle sue dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.13.0	Flessibili

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.12.0	Flessibili

2.1.5

La tabella seguente elenca le dipendenze per la versione 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.11.0	Flessibili

2.1.4

La tabella seguente elenca le dipendenze per la versione 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.10.0	Flessibili

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.9.0	Flessibili

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.8.0	Flessibili

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.7.0	Flessibili

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.6.0	Flessibili

2.0.9

La tabella seguente elenca le dipendenze per la versione 2.0.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili

2.0.8

La tabella seguente elenca le dipendenze per la versione 2.0.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili

2.0.7

La tabella seguente elenca le dipendenze per la versione 2.0.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

2.0.6

La tabella seguente elenca le dipendenze per la versione 2.0.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili

2.0.4 and 2.0.5

La tabella seguente elenca le dipendenze per le versioni 2.0.4 e 2.0.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`cloudSecrets`

Un elenco di segreti di Secrets Manager da distribuire sul dispositivo principale. È possibile specificare etichette per definire quali versioni di ogni segreto distribuire. Se non specifichi una versione, questo componente distribuisce la versione con l'etichetta di staging allegata. `AWSCURRENT` Per ulteriori informazioni, consulta Staging labels nella Guida per l'AWS Secrets Managerutente.

Il componente secret manager memorizza i segreti nella cache locale. Se il valore segreto cambia in Secrets Manager, questo componente non recupera automaticamente il nuovo valore. Per aggiornare la copia locale, assegna al segreto una nuova etichetta e configura questo componente per recuperare il segreto identificato dalla nuova etichetta.

Ogni oggetto contiene le seguenti informazioni:

`arn`

L'ARN del segreto da implementare. L'ARN del segreto può essere un ARN completo o un ARN parziale. Si consiglia di specificare un ARN completo anziché un ARN parziale. Per ulteriori informazioni, vedere Ricerca di un segreto da un ARN parziale. Di seguito è riportato un esempio di ARN completo e ARN parziale:

- ARN completo: `arn:aws:secretsmanager:us-east-2:111122223333:secret:`*`SecretName`*`-abcdef`
- ARN parziale: `arn:aws:secretsmanager:us-east-2:111122223333:secret:`*`SecretName`*

`labels`

(Facoltativo) Un elenco di etichette per identificare le versioni del segreto da distribuire sul dispositivo principale.

Ogni etichetta deve essere una stringa.

Example Esempio: configurazione, unione, aggiornamento

```
{
  "cloudSecrets": [
    {
      "arn": "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyGreengrassSecret-
abcdef"
    }
  ]
}
```

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.1.7	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.5	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.4	Correzioni di bug e miglioramenti Risolve un problema per cui i segreti memorizzati nella cache venivano rimossi quando il gestore segreto veniva distribuito e Greengrass nucleus si riavviava. Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.1	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.0	Nuove funzionalità • Aggiunge il supporto per l'integrazione della sicurezza hardware. Il componente secret manager può crittografare e decrittografare i segreti utilizzando una chiave privata archiviata in un modulo di sicurezza hardware (HSM). Per ulteriori informazioni, consulta Integrazione della sicurezza hardware. Correzioni di bug e miglioramenti • Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.

Versione	Modifiche
2.0.9	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.8	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.0.7	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.0.6	Versione aggiornata per la versione 2.1.0 di Greengrass nucleus.
2.0.5	Miglioramenti • Aggiunge il supporto per le regioni e le regioni AWS della Cina. AWS GovCloud (US)
2.0.4	Versione iniziale.

Tunneling sicuro

Con il `aws.greengrass.SecureTunneling` componente, è possibile stabilire una comunicazione bidirezionale sicura con un dispositivo principale Greengrass situato dietro firewall limitati.

Ad esempio, immagina di avere un dispositivo core Greengrass protetto da un firewall che vieta tutte le connessioni in entrata. Il tunneling sicuro utilizza MQTT per trasferire un token di accesso al dispositivo e quindi lo utilizza WebSockets per stabilire una connessione SSH al dispositivo tramite il firewall. Con questo tunnel AWS IoT gestito, puoi aprire la connessione SSH necessaria per il tuo dispositivo. Per ulteriori informazioni sull'utilizzo del tunneling AWS IoT sicuro per la connessione a dispositivi remoti, consulta AWS IoTSecure Tunneling nella Developer Guide. AWS IoT

Questo componente sottoscrive il broker di messaggi AWS IoT Core MQTT sull'`$aws/things/`*`greengrass-core-device`*`/tunnels/notify`argomento per ricevere notifiche di tunneling sicuro.

Argomenti

- Versioni
- Type
- Sistema operativo
- Requisiti

- Dipendenze
- Configurazione
- File di registro locale
- Licenze
- Utilizzo
- Consulta anche
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 1.0.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato solo sui dispositivi principali di Linux.

Architetture:

- Arm v71
- Armv8 (AArch64)
- x86_64

Requisiti

Questo componente ha i seguenti requisiti:

- Almeno 32 MB di spazio su disco disponibile per il componente di tunneling sicuro. Questo requisito non include il software di base Greengrass o altri componenti in esecuzione sullo stesso dispositivo.
- Sono disponibili almeno 16 MB di RAM per il componente di tunneling sicuro. Questo requisito non include il software di base Greengrass o altri componenti in esecuzione sullo stesso dispositivo. Per ulteriori informazioni, consulta Controlla l'allocazione della memoria con le opzioni JVM.
- La versione 2.25 o superiore della GNU C Library (glibc) con un kernel Linux 3.2 o superiore è richiesta per la versione 1.0.12 e successive del componente di tunneling sicuro. Le versioni del sistema operativo e delle librerie che hanno superato la data di scadenza del supporto a lungo termine non sono supportate. È necessario utilizzare un sistema operativo e delle librerie con supporto a lungo termine.
- Sia il sistema operativo che il runtime Java devono essere installati a 64 bit.
- Python 3.5 o successivo installato sul dispositivo principale Greengrass e aggiunto alla variabile di ambiente PATH.
- `libcrypto.so.1.1`installato sul dispositivo principale Greengrass e aggiunto alla variabile di ambiente PATH.
- Apri il traffico in uscita sulla porta 443 del dispositivo principale Greengrass.
- Attiva il supporto per il servizio di comunicazione che desideri utilizzare per comunicare con il dispositivo principale Greengrass. Ad esempio, per aprire una connessione SSH al dispositivo, è necessario attivare SSH su quel dispositivo.

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`data.tunneling.iot.` *`region`*`.amazonaws.com`	443	Sì	Stabilisci tunnel sicuri.

Dipendenze

Quando distribuisci un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle sue dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

1.0.18

La tabella seguente elenca le dipendenze per la versione 1.0.18 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili

1.0.16 – 1.0.17

La tabella seguente elenca le dipendenze per le versioni da 1.0.16 a 1.0.17 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili

1.0.14 – 1.0.15

La tabella seguente elenca le dipendenze per le versioni da 1.0.14 a 1.0.15 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili

1.0.11 – 1.0.13

La tabella seguente elenca le dipendenze per le versioni 1.0.11 - 1.0.13 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili

1.0.10

La tabella seguente elenca le dipendenze per la versione 1.0.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili

1.0.9

La tabella seguente elenca le dipendenze per la versione 1.0.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili

1.0.8

La tabella seguente elenca le dipendenze per la versione 1.0.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili

1.0.5 - 1.0.7

La tabella seguente elenca le dipendenze per le versioni da 1.0.5 a 1.0.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili

1.0.4

La tabella seguente elenca le dipendenze per la versione 1.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili

1.0.3

La tabella seguente elenca le dipendenze per la versione 1.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili

1.0.2

La tabella seguente elenca le dipendenze per la versione 1.0.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

1.0.1

La tabella seguente elenca le dipendenze per la versione 1.0.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili

1.0.0

La tabella seguente elenca le dipendenze per la versione 1.0.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`OS_DIST_INFO`

(Facoltativo) Il sistema operativo del dispositivo principale. Per impostazione predefinita, il componente tenta di identificare automaticamente il sistema operativo in esecuzione sul dispositivo principale. Se il componente non si avvia con il valore predefinito, utilizzate questo valore per specificare il sistema operativo. Per un elenco dei sistemi operativi supportati per questo componente, vedereRequisiti per il dispositivo.

Questo valore può essere uno dei seguenti:`auto`,`ubuntu`,`amzn2`,`raspberrypi`.

Impostazione predefinita: `auto`

`accessControl`

(Facoltativo) L'oggetto che contiene la politica di autorizzazione che consente al componente di sottoscrivere l'argomento delle notifiche di tunneling sicuro.

> **Note**
>
> Non modificate questo parametro di configurazione se la distribuzione è destinata a un gruppo di oggetti. Se la distribuzione riguarda un singolo dispositivo core e desideri limitarne l'abbonamento all'argomento del dispositivo, specifica il nome dell'oggetto del dispositivo principale. Nel `resources` valore della politica di autorizzazione del

dispositivo, sostituisci la wildcard dell'argomento MQTT con il nome dell'oggetto del dispositivo.

```
{
  "aws.greengrass.ipc.mqttproxy": {
    "aws.iot.SecureTunneling:mqttproxy:1": {
      "policyDescription": "Access to tunnel notification pubsub topic",
      "operations": [
        "aws.greengrass#SubscribeToIoTCore"
      ],
      "resources": [
        "$aws/things/+/tunnels/notify"
      ]
    }
  }
}
```

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica di consentire a questo componente di aprire tunnel sicuri su un dispositivo principale denominato **`MyGreengrassCore`** che esegue Ubuntu.

```
{
  "OS_DIST_INFO": "ubuntu",
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "aws.iot.SecureTunneling:mqttproxy:1": {
        "policyDescription": "Access to tunnel notification pubsub topic",
        "operations": [
          "aws.greengrass#SubscribeToIoTCore"
        ],
        "resources": [
          "$aws/things/MyGreengrassCore/tunnels/notify"
        ]
      }
    }
  }
}
```

File di registro locale

Questo componente utilizza il seguente file di registro.

```
/greengrass/v2/logs/aws.greengrass.SecureTunneling.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale.

  ```
  sudo tail -f /greengrass/v2/logs/aws.greengrass.SecureTunneling.log
  ```

Licenze

Questo componente include i seguenti software/licenze di terze parti:

- AWS IoTDevice Client /Apache License 2.0
- SDK per dispositivi AWS IoT per Java/Apache License 2.0
- Licenza gson/Apache 2.0
- log4j /Apache Licenza 2.0
- slf4j /Apache Licenza 2.0

Utilizzo

Per utilizzare il componente di tunneling sicuro sul tuo dispositivo, procedi come segue:

1. Implementa il componente di tunneling sicuro sul tuo dispositivo.
2. Apri la AWS IoT console. Dal menu a sinistra, scegli Azioni remote, quindi scegli Tunnel sicuri.
3. Crea un tunnel per il tuo dispositivo Greengrass.
4. Scarica il token di accesso sorgente.
5. Usa il proxy locale con il token di accesso sorgente per connetterti alla tua destinazione. Per ulteriori informazioni, consulta Come utilizzare il proxy locale nella Guida per gli AWS IoT sviluppatori.

Consulta anche

- AWS IoTtunneling sicuro nella Guida per gli sviluppatori AWS IoT
- Come usare il proxy locale nella Developer Guide AWS IoT

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
1.0.18	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
1.0.17	Correzioni di bug e miglioramenti • Risolve il problema di pulizia dei thread che impediva agli utenti di creare tunnel. Questo componente ora pulirà un thread una volta ricevuto il CloseTunnel segnale o se il tunnel è scaduto dopo 12 ore.
1.0.16	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
1.0.15	Correzioni di bug e miglioramenti • Risolve un problema di avvio per gli utenti che non dispongono di una home directory sul dispositivo. Il componente Secure Tunneling ora si avvia senza creare una directory per i documenti shadow.
1.0.14	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
1.0.13	Correzioni di bug e miglioramenti • Risolve un problema per cui un processo client orfano impedisce a più di un tunnel di indirizzare il dispositivo.
1.0.12	Correzioni di bug e miglioramenti • Aggiunge il supporto per x86_64 (AMD64) e ARMv8 (Aarch64) durante l'esecuzione su sistema operativo Raspberry Pi.
1.0.11	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
1.0.10	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.

Versione	Modifiche
1.0.9	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
1.0.8	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
1.0.7	Correzioni di bug e miglioramenti • Risolve un problema a causa del quale il componente si disconnette quando si trasferiscono file di grandi dimensioni tramite SCP.
1.0.6	Questa versione contiene correzioni di bug.
1.0.5	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
1.0.4	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
1.0.3	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
1.0.2	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
1.0.1	Versione aggiornata per la versione 2.1.0 di Greengrass nucleus.
1.0.0	Versione iniziale.

Gestore delle ombre

Il componente shadow manager (`aws.greengrass.ShadowManager`) abilita il servizio shadow locale sul dispositivo principale. Il servizio shadow locale consente ai componenti di utilizzare la comunicazione tra processi per interagire con le ombre locali. Il componente shadow manager gestisce l'archiviazione dei documenti shadow locali e gestisce anche la sincronizzazione degli stati shadow locali con il servizio AWS IoT Device Shadow.

Per ulteriori informazioni su come i dispositivi core Greengrass possono interagire con le ombre, vedere. Interagisci con le ombre dei dispositivi

Argomenti

- Versioni
- Type
- Sistema operativo

- Requisiti
- Dipendenze
- Configurazione
- File di registro locale
- Changelog

Versioni

Questo componente ha le seguenti versioni:

- 2.3.x
- 2.2.x
- 2.1.x
- 2,0x

Type

Questo componente è un componente del plugin ()`aws.greengrass.plugin`. Il nucleo Greengrass esegue questo componente nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di questo componente sul dispositivo principale.

Questo componente utilizza lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Per ulteriori informazioni, consultare Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- (Facoltativo) Per sincronizzare le ombre con il servizio AWS IoT Device Shadow, la policy AWS IoT del dispositivo principale di Greengrass deve consentire le AWS IoT Core seguenti azioni di policy ombra:
  - `iot:GetThingShadow`
  - `iot:UpdateThingShadow`
  - `iot:DeleteThingShadow`

  Per ulteriori informazioni su queste AWS IoT Core politiche, consulta le azioni AWS IoT Core politiche nella Guida per gli AWS IoT sviluppatori.

  Per ulteriori informazioni sulla AWS IoT politica minima, vedere AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali
- Il componente shadow manager è supportato per l'esecuzione in un VPC.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.3.5 – 2.3.7

La tabella seguente elenca le dipendenze per le versioni da 2.3.5 a 2.3.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.13.0	Flessibili

2.3.3 and 2.3.4

La tabella seguente elenca le dipendenze per le versioni 2.3.3 e 2.3.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.12.0	Flessibili

2.3.2

La tabella seguente elenca le dipendenze per la versione 2.3.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.11.0	Flessibili

2.3.0 and 2.3.1

La tabella seguente elenca le dipendenze per le versioni 2.3.0 e 2.3.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.5.0 <2.10.0	Flessibili

2.2.3 and 2.2.4

La tabella seguente elenca le dipendenze per le versioni 2.2.3 e 2.2.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <3.0.0	Flessibili

2.2.2

La tabella seguente elenca le dipendenze per la versione 2.2.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.9.0	Flessibili

2.2.1

La tabella seguente elenca le dipendenze per la versione 2.2.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.8.0	Flessibili

2.1.1 and 2.2.0

La tabella seguente elenca le dipendenze per le versioni 2.1.1 e 2.2.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.7.0	Flessibili

2.0.5 - 2.1.0

La tabella seguente elenca le dipendenze per le versioni da 2.0.5 a 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.6.0	Flessibili

2.0.3 and 2.0.4

La tabella seguente elenca le dipendenze per le versioni 2.0.3 e 2.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.5.0	Flessibili

2.0.1 and 2.0.2

La tabella seguente elenca le dipendenze per le versioni 2.0.1 e 2.0.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.4.0	Flessibili

2.0.0

La tabella seguente elenca le dipendenze per la versione 2.0.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.2.0 <2.3.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

2.3.x

`strategy`

(Facoltativo) La strategia utilizzata da questo componente per sincronizzare le ombre tra AWS IoT Core e il dispositivo principale.

Questo oggetto contiene le seguenti informazioni.

`type`

(Facoltativo) Il tipo di strategia utilizzato da questo componente per sincronizzare le ombre tra AWS IoT Core e il dispositivo principale. Seleziona una delle opzioni seguenti:

- `realTime`— Sincronizza le ombre AWS IoT Core ogni volta che si verifica un aggiornamento delle ombre.
- `periodic`— Sincronizza le ombre con AWS IoT Core un intervallo regolare specificato con il `delay` parametro di configurazione.

Impostazione predefinita: `realTime`

`delay`

(Facoltativo) L'intervallo in secondi con cui questo componente sincronizza le ombre AWS IoT Core, quando si specifica la strategia di sincronizzazione. `periodic`

> **Note**
>
> Questo parametro è obbligatorio se si specifica la strategia di sincronizzazione. `periodic`

`synchronize`

(Facoltativo) Le impostazioni di sincronizzazione che determinano il modo in cui le ombre vengono sincronizzate con. Cloud AWS

> **Note**
>
> È necessario creare un aggiornamento della configurazione con questa proprietà per sincronizzare le ombre con. Cloud AWS

Questo oggetto contiene le seguenti informazioni.

`coreThing`

(Facoltativo) Le ombre del dispositivo principale da sincronizzare. Questo oggetto contiene le seguenti informazioni.

`classic`

(Facoltativo) Per impostazione predefinita, lo shadow manager sincronizza lo stato locale dello shadow classico per il dispositivo principale con. Cloud AWS Se non vuoi sincronizzare la classica ombra del dispositivo, impostala su. `false`

Impostazione predefinita: `true`

`namedShadows`

(Facoltativo) L'elenco delle ombre dei dispositivi principali denominate da sincronizzare. È necessario specificare i nomi esatti delle ombre.

> **Warning**
>
> Il AWS IoT Greengrass servizio utilizza lo shadow `AWSManagedGreengrassV2Deployment` denominato per gestire le distribuzioni destinate a singoli dispositivi core. Questa ombra denominata è riservata all'uso da parte del AWS IoT Greengrass servizio. Non aggiornare o eliminare questa ombra denominata.

`shadowDocumentsMap`

(Facoltativo) Le ombre aggiuntive del dispositivo da sincronizzare. L'utilizzo di questo parametro di configurazione semplifica la specificazione dei documenti shadow. Si consiglia di utilizzare questo parametro al posto dell'`shadowDocuments`oggetto.

> **Note**
>
> Se si specifica un `shadowDocumentsMap` oggetto, non è necessario specificare un `shadowDocuments` oggetto.

Ogni oggetto contiene le seguenti informazioni:

`thingName`

La configurazione shadow per *`ThingName`* per questa configurazione shadow.

`classic`

(Facoltativo) Se non desiderate sincronizzare la classica ombra del dispositivo per il `thingName` dispositivo, impostatela `false` su.

`namedShadows`

L'elenco delle ombre denominate che desideri sincronizzare. È necessario specificare i nomi esatti delle ombre.

`shadowDocuments`

(Facoltativo) L'elenco delle ombre aggiuntive del dispositivo da sincronizzare. Ti consigliamo invece di utilizzare il `shadowDocumentsMap` parametro.

> **Note**
>
> Se si specifica un `shadowDocuments` oggetto, non è necessario specificare un `shadowDocumentsMap` oggetto.

Ogni oggetto in questo elenco contiene le seguenti informazioni.

`thingName`

Il nome dell'oggetto del dispositivo per cui sincronizzare le ombre.

`classic`

(Facoltativo) Se non desideri sincronizzare la classica ombra del dispositivo per il `thingName` dispositivo, impostala su. `false`

Impostazione predefinita: `true`

`namedShadows`

(Facoltativo) L'elenco delle ombre dei dispositivi denominate che desideri sincronizzare. È necessario specificare i nomi esatti delle ombre.

`direction`

(Facoltativo) La direzione in cui sincronizzare le ombre tra il servizio shadow locale e il. Cloud AWSÈ possibile configurare questa opzione per ridurre la larghezza di banda e le connessioni a. Cloud AWS Seleziona una delle opzioni seguenti:

- `betweenDeviceAndCloud`— Sincronizza le ombre tra il servizio shadow locale e il. Cloud AWS
- `deviceToCloud`— Invia gli aggiornamenti shadow dal servizio shadow locale al e ignora gli aggiornamenti shadow provenienti da. Cloud AWS Cloud AWS
- `cloudToDevice`— Ricevi aggiornamenti shadow dal Cloud AWS servizio shadow locale e non inviali dal servizio shadow locale a Cloud AWS.

Impostazione predefinita: `BETWEEN_DEVICE_AND_CLOUD`

`rateLimits`

(Facoltativo) Le impostazioni che determinano i limiti di velocità per le richieste del servizio shadow.

Questo oggetto contiene le seguenti informazioni.

`maxOutboundSyncUpdatesPerSecond`

(Facoltativo) Il numero massimo di richieste di sincronizzazione al secondo trasmesse dal dispositivo.

Impostazione predefinita: 100 richieste/secondo

`maxTotalLocalRequestsRate`

(Facoltativo) Il numero massimo di richieste IPC locali al secondo inviate al dispositivo principale.

Impostazione predefinita: 200 richieste/secondo

`maxLocalRequestsPerSecondPerThing`

(Facoltativo) Il numero massimo di richieste IPC locali al secondo inviate per ogni oggetto IoT connesso.

Impostazione predefinita: 20 richieste/secondo per ogni cosa

> **Note**
>
> Questi parametri relativi ai limiti di velocità definiscono il numero massimo di richieste al secondo per il servizio shadow locale. Il numero massimo di richieste al secondo per il servizio AWS IoT Device Shadow dipende dal tuo Regione AWS. Per ulteriori informazioni, consulta i limiti per l'API AWS IoT Device Shadow Service in Riferimenti generali di Amazon Web Services.

`shadowDocumentSizeLimitBytes`

(Facoltativo) La dimensione massima consentita di ogni documento di stato JSON per le ombre locali.

Se si aumenta questo valore, è necessario aumentare anche il limite di risorse per il documento di stato JSON per le ombre nel cloud. Per ulteriori informazioni, consulta i limiti per l'API AWS IoT Device Shadow Service in Riferimenti generali di Amazon Web Services.

Impostazione predefinita: 8192 byte

Massimo: 30720 byte

Example Esempio: aggiornamento di fusione della configurazione

L'esempio seguente mostra un esempio di aggiornamento di fusione della configurazione con tutti i parametri di configurazione disponibili per il componente shadow manager.

```
{
   "strategy":{
      "type":"periodic",
      "delay":300
   },
   "synchronize":{
      "shadowDocumentsMap":{
         "MyDevice1":{
            "classic":false,
            "namedShadows":[
               "MyShadowA",
               "MyShadowB"
            ]
         },
         "MyDevice2":{
            "classic":true,
            "namedShadows":[]
         }
      },
      "direction":"betweenDeviceAndCloud"
   },
   "rateLimits":{
      "maxOutboundSyncUpdatesPerSecond":100,
      "maxTotalLocalRequestsRate":200,
      "maxLocalRequestsPerSecondPerThing":20
   },
   "shadowDocumentSizeLimitBytes":8192
}
```

2.2.x

`strategy`

(Facoltativo) La strategia utilizzata da questo componente per sincronizzare le ombre tra AWS IoT Core e il dispositivo principale.

Questo oggetto contiene le seguenti informazioni.

`type`

(Facoltativo) Il tipo di strategia utilizzato da questo componente per sincronizzare le ombre tra AWS IoT Core e il dispositivo principale. Seleziona una delle opzioni seguenti:

- `realTime`— Sincronizza le ombre AWS IoT Core ogni volta che si verifica un aggiornamento delle ombre.
- `periodic`— Sincronizza le ombre con AWS IoT Core un intervallo regolare specificato con il `delay` parametro di configurazione.

Impostazione predefinita: `realTime`

`delay`

(Facoltativo) L'intervallo in secondi con cui questo componente sincronizza le ombre AWS IoT Core, quando si specifica la strategia di sincronizzazione. `periodic`

> **Note**
>
> Questo parametro è obbligatorio se si specifica la strategia di sincronizzazione. `periodic`

`synchronize`

(Facoltativo) Le impostazioni di sincronizzazione che determinano il modo in cui le ombre vengono sincronizzate con. Cloud AWS

> **Note**
>
> È necessario creare un aggiornamento della configurazione con questa proprietà per sincronizzare le ombre con. Cloud AWS

Questo oggetto contiene le seguenti informazioni.

`coreThing`

(Facoltativo) Le ombre del dispositivo principale da sincronizzare. Questo oggetto contiene le seguenti informazioni.

`classic`

(Facoltativo) Per impostazione predefinita, lo shadow manager sincronizza lo stato locale dello shadow classico per il dispositivo principale con. Cloud AWS Se non vuoi sincronizzare la classica ombra del dispositivo, impostala su. `false`

Impostazione predefinita: `true`

`namedShadows`

(Facoltativo) L'elenco delle ombre dei dispositivi principali denominate da sincronizzare. È necessario specificare i nomi esatti delle ombre.

> **Warning**
>
> Il AWS IoT Greengrass servizio utilizza lo shadow `AWSManagedGreengrassV2Deployment` denominato per gestire le distribuzioni destinate a singoli dispositivi core. Questa ombra denominata è riservata all'uso da parte del AWS IoT Greengrass servizio. Non aggiornare o eliminare questa ombra denominata.

`shadowDocumentsMap`

(Facoltativo) Le ombre aggiuntive del dispositivo da sincronizzare. L'utilizzo di questo parametro di configurazione semplifica la specificazione dei documenti shadow. Si consiglia di utilizzare questo parametro al posto dell'`shadowDocuments`oggetto.

> **Note**
>
> Se si specifica un `shadowDocumentsMap` oggetto, non è necessario specificare un `shadowDocuments` oggetto.

Ogni oggetto contiene le seguenti informazioni:

`thingName`

La configurazione shadow per *`ThingName`* per questa configurazione shadow.

`classic`

(Facoltativo) Se non desiderate sincronizzare la classica ombra del dispositivo per il `thingName` dispositivo, impostatela `false` su.

`namedShadows`

L'elenco delle ombre denominate che desideri sincronizzare. È necessario specificare i nomi esatti delle ombre.

`shadowDocuments`

(Facoltativo) L'elenco delle ombre aggiuntive del dispositivo da sincronizzare. Ti consigliamo invece di utilizzare il `shadowDocumentsMap` parametro.

> **Note**
>
> Se si specifica un `shadowDocuments` oggetto, non è necessario specificare un `shadowDocumentsMap` oggetto.

Ogni oggetto in questo elenco contiene le seguenti informazioni.

`thingName`

Il nome dell'oggetto del dispositivo per cui sincronizzare le ombre.

`classic`

(Facoltativo) Se non desideri sincronizzare la classica ombra del dispositivo per il `thingName` dispositivo, impostala su. `false`

Impostazione predefinita: `true`

`namedShadows`

(Facoltativo) L'elenco delle ombre dei dispositivi denominate che desideri sincronizzare. È necessario specificare i nomi esatti delle ombre.

`direction`

(Facoltativo) La direzione in cui sincronizzare le ombre tra il servizio shadow locale e il. Cloud AWSÈ possibile configurare questa opzione per ridurre la larghezza di banda e le connessioni a. Cloud AWS Seleziona una delle opzioni seguenti:

- `betweenDeviceAndCloud`— Sincronizza le ombre tra il servizio shadow locale e il. Cloud AWS
- `deviceToCloud`— Invia gli aggiornamenti shadow dal servizio shadow locale al e ignora gli aggiornamenti shadow provenienti da. Cloud AWS Cloud AWS
- `cloudToDevice`— Ricevi aggiornamenti shadow dal Cloud AWS servizio shadow locale e non inviali dal servizio shadow locale a Cloud AWS.

Impostazione predefinita: `BETWEEN_DEVICE_AND_CLOUD`

`rateLimits`

(Facoltativo) Le impostazioni che determinano i limiti di velocità per le richieste del servizio shadow.

Questo oggetto contiene le seguenti informazioni.

`maxOutboundSyncUpdatesPerSecond`

(Facoltativo) Il numero massimo di richieste di sincronizzazione al secondo trasmesse dal dispositivo.

Impostazione predefinita: 100 richieste/secondo

`maxTotalLocalRequestsRate`

(Facoltativo) Il numero massimo di richieste IPC locali al secondo inviate al dispositivo principale.

Impostazione predefinita: 200 richieste/secondo

`maxLocalRequestsPerSecondPerThing`

(Facoltativo) Il numero massimo di richieste IPC locali al secondo inviate per ogni oggetto IoT connesso.

Impostazione predefinita: 20 richieste/secondo per ogni cosa

> **Note**
>
> Questi parametri relativi ai limiti di velocità definiscono il numero massimo di richieste al secondo per il servizio shadow locale. Il numero massimo di richieste al secondo

per il servizio AWS IoT Device Shadow dipende dal tuo Regione AWS. Per ulteriori informazioni, consulta i limiti per l'API AWS IoT Device Shadow Service in Riferimenti generali di Amazon Web Services.

`shadowDocumentSizeLimitBytes`

(Facoltativo) La dimensione massima consentita di ogni documento di stato JSON per le ombre locali.

Se si aumenta questo valore, è necessario aumentare anche il limite di risorse per il documento di stato JSON per le ombre nel cloud. Per ulteriori informazioni, consulta i limiti per l'API AWS IoT Device Shadow Service in Riferimenti generali di Amazon Web Services.

Impostazione predefinita: 8192 byte

Massimo: 30720 byte

Example Esempio: aggiornamento di fusione della configurazione

L'esempio seguente mostra un esempio di aggiornamento di fusione della configurazione con tutti i parametri di configurazione disponibili per il componente shadow manager.

```
{
   "strategy":{
      "type":"periodic",
      "delay":300
   },
   "synchronize":{
      "shadowDocumentsMap":{
         "MyDevice1":{
            "classic":false,
            "namedShadows":[
               "MyShadowA",
               "MyShadowB"
            ]
         },
         "MyDevice2":{
            "classic":true,
            "namedShadows":[]
         }
```

```
      },
      "direction":"betweenDeviceAndCloud"
   },
   "rateLimits":{
      "maxOutboundSyncUpdatesPerSecond":100,
      "maxTotalLocalRequestsRate":200,
      "maxLocalRequestsPerSecondPerThing":20
   },
   "shadowDocumentSizeLimitBytes":8192
}
```

2.1.x

`strategy`

(Facoltativo) La strategia utilizzata da questo componente per sincronizzare le ombre tra AWS IoT Core e il dispositivo principale.

Questo oggetto contiene le seguenti informazioni.

`type`

(Facoltativo) Il tipo di strategia utilizzato da questo componente per sincronizzare le ombre tra AWS IoT Core e il dispositivo principale. Seleziona una delle opzioni seguenti:

- `realTime`— Sincronizza le ombre AWS IoT Core ogni volta che si verifica un aggiornamento delle ombre.
- `periodic`— Sincronizza le ombre con AWS IoT Core un intervallo regolare specificato con il `delay` parametro di configurazione.

Impostazione predefinita: `realTime`

`delay`

(Facoltativo) L'intervallo in secondi con cui questo componente sincronizza le ombre AWS IoT Core, quando si specifica la strategia di sincronizzazione. `periodic`

> **Note**
>
> Questo parametro è obbligatorio se si specifica la strategia di sincronizzazione. `periodic`

`synchronize`

(Facoltativo) Le impostazioni di sincronizzazione che determinano il modo in cui le ombre vengono sincronizzate con. Cloud AWS

> **Note**
>
> È necessario creare un aggiornamento della configurazione con questa proprietà per sincronizzare le ombre con. Cloud AWS

Questo oggetto contiene le seguenti informazioni.

`coreThing`

(Facoltativo) Le ombre del dispositivo principale da sincronizzare. Questo oggetto contiene le seguenti informazioni.

`classic`

(Facoltativo) Per impostazione predefinita, lo shadow manager sincronizza lo stato locale dello shadow classico per il dispositivo principale con. Cloud AWS Se non vuoi sincronizzare la classica ombra del dispositivo, impostala su. `false`

Impostazione predefinita: `true`

`namedShadows`

(Facoltativo) L'elenco delle ombre dei dispositivi principali denominate da sincronizzare. È necessario specificare i nomi esatti delle ombre.

> **Warning**
>
> Il AWS IoT Greengrass servizio utilizza lo shadow `AWSManagedGreengrassV2Deployment` denominato per gestire le distribuzioni destinate a singoli dispositivi core. Questa ombra denominata è riservata all'uso da parte del AWS IoT Greengrass servizio. Non aggiornare o eliminare questa ombra denominata.

`shadowDocumentsMap`

(Facoltativo) Le ombre aggiuntive del dispositivo da sincronizzare. L'utilizzo di questo parametro di configurazione semplifica la specificazione dei documenti shadow. Si consiglia di utilizzare questo parametro al posto dell'`shadowDocuments`oggetto.

> **Note**
>
> Se si specifica un `shadowDocumentsMap` oggetto, non è necessario specificare un `shadowDocuments` oggetto.

Ogni oggetto contiene le seguenti informazioni:

`thingName`

La configurazione shadow per *`ThingName`* per questa configurazione shadow.

`classic`

(Facoltativo) Se non desiderate sincronizzare la classica ombra del dispositivo per il `thingName` dispositivo, impostatela `false` su.

`namedShadows`

L'elenco delle ombre denominate che desideri sincronizzare. È necessario specificare i nomi esatti delle ombre.

`shadowDocuments`

(Facoltativo) L'elenco delle ombre aggiuntive del dispositivo da sincronizzare. Ti consigliamo invece di utilizzare il `shadowDocumentsMap` parametro.

> **Note**
>
> Se si specifica un `shadowDocuments` oggetto, non è necessario specificare un `shadowDocumentsMap` oggetto.

Ogni oggetto in questo elenco contiene le seguenti informazioni.

`thingName`

Il nome dell'oggetto del dispositivo per cui sincronizzare le ombre.

`classic`

(Facoltativo) Se non desideri sincronizzare la classica ombra del dispositivo per il `thingName` dispositivo, impostala su. `false`

Impostazione predefinita: `true`

`namedShadows`

(Facoltativo) L'elenco delle ombre dei dispositivi denominate che desideri sincronizzare. È necessario specificare i nomi esatti delle ombre.

`rateLimits`

(Facoltativo) Le impostazioni che determinano i limiti di velocità per le richieste di servizi shadow.

Questo oggetto contiene le seguenti informazioni.

`maxOutboundSyncUpdatesPerSecond`

(Facoltativo) Il numero massimo di richieste di sincronizzazione al secondo trasmesse dal dispositivo.

Impostazione predefinita: 100 richieste/secondo

`maxTotalLocalRequestsRate`

(Facoltativo) Il numero massimo di richieste IPC locali al secondo inviate al dispositivo principale.

Impostazione predefinita: 200 richieste/secondo

`maxLocalRequestsPerSecondPerThing`

(Facoltativo) Il numero massimo di richieste IPC locali al secondo inviate per ogni oggetto IoT connesso.

Impostazione predefinita: 20 richieste/secondo per ogni cosa

> **Note**
>
> Questi parametri relativi ai limiti di velocità definiscono il numero massimo di richieste al secondo per il servizio shadow locale. Il numero massimo di richieste al secondo

per il servizio AWS IoT Device Shadow dipende dal tuo Regione AWS. Per ulteriori informazioni, consulta i limiti per l'API AWS IoT Device Shadow Service in Riferimenti generali di Amazon Web Services.

`shadowDocumentSizeLimitBytes`

(Facoltativo) La dimensione massima consentita di ogni documento di stato JSON per le ombre locali.

Se si aumenta questo valore, è necessario aumentare anche il limite di risorse per il documento di stato JSON per le ombre nel cloud. Per ulteriori informazioni, consulta i limiti per l'API AWS IoT Device Shadow Service in Riferimenti generali di Amazon Web Services.

Impostazione predefinita: 8192 byte

Massimo: 30720 byte

Example Esempio: aggiornamento di fusione della configurazione

L'esempio seguente mostra un esempio di aggiornamento di fusione della configurazione con tutti i parametri di configurazione disponibili per il componente shadow manager.

```
{
   "strategy":{
      "type":"periodic",
      "delay":300
   },
   "synchronize":{
      "shadowDocumentsMap":{
         "MyDevice1":{
            "classic":false,
            "namedShadows":[
               "MyShadowA",
               "MyShadowB"
            ]
         },
         "MyDevice2":{
            "classic":true,
            "namedShadows":[]
         }
```

```
      },
      "direction":"betweenDeviceAndCloud"
   },
   "rateLimits":{
      "maxOutboundSyncUpdatesPerSecond":100,
      "maxTotalLocalRequestsRate":200,
      "maxLocalRequestsPerSecondPerThing":20
   },
   "shadowDocumentSizeLimitBytes":8192
}
```

2.0.x

`synchronize`

(Facoltativo) Le impostazioni di sincronizzazione che determinano il modo in cui le ombre vengono sincronizzate con. Cloud AWS

> **Note**
>
> È necessario creare un aggiornamento della configurazione con questa proprietà per sincronizzare le ombre con. Cloud AWS

Questo oggetto contiene le seguenti informazioni.

`coreThing`

(Facoltativo) Le ombre del dispositivo principale da sincronizzare. Questo oggetto contiene le seguenti informazioni.

`classic`

(Facoltativo) Per impostazione predefinita, lo shadow manager sincronizza lo stato locale dello shadow classico per il dispositivo principale con. Cloud AWS Se non vuoi sincronizzare la classica ombra del dispositivo, impostala su. `false`

Impostazione predefinita: `true`

`namedShadows`

(Facoltativo) L'elenco delle ombre dei dispositivi principali denominate da sincronizzare. È necessario specificare i nomi esatti delle ombre.

> **Warning**
>
> Il AWS IoT Greengrass servizio utilizza lo shadow `AWSManagedGreengrassV2Deployment` denominato per gestire le distribuzioni destinate a singoli dispositivi core. Questa ombra denominata è riservata all'uso da parte del AWS IoT Greengrass servizio. Non aggiornare o eliminare questa ombra denominata.

`shadowDocumentsMap`

(Facoltativo) Le ombre aggiuntive del dispositivo da sincronizzare. L'utilizzo di questo parametro di configurazione semplifica la specificazione dei documenti shadow. Si consiglia di utilizzare questo parametro al posto dell'`shadowDocuments`oggetto.

> **Note**
>
> Se si specifica un `shadowDocumentsMap` oggetto, non è necessario specificare un `shadowDocuments` oggetto.

Ogni oggetto contiene le seguenti informazioni:

`thingName`

La configurazione shadow per *`ThingName`* per questa configurazione shadow.

`classic`

(Facoltativo) Se non desiderate sincronizzare la classica ombra del dispositivo per il `thingName` dispositivo, impostatela `false` su.

`namedShadows`

L'elenco delle ombre denominate che desideri sincronizzare. È necessario specificare i nomi esatti delle ombre.

`shadowDocuments`

(Facoltativo) L'elenco delle ombre aggiuntive del dispositivo da sincronizzare. Ti consigliamo invece di utilizzare il `shadowDocumentsMap` parametro.

> **Note**
>
> Se si specifica un `shadowDocuments` oggetto, non è necessario specificare un `shadowDocumentsMap` oggetto.

Ogni oggetto in questo elenco contiene le seguenti informazioni.

`thingName`

Il nome dell'oggetto del dispositivo per cui sincronizzare le ombre.

`classic`

(Facoltativo) Se non desideri sincronizzare la classica ombra del dispositivo per il `thingName` dispositivo, impostala su. `false`

Impostazione predefinita: `true`

`namedShadows`

(Facoltativo) L'elenco delle ombre dei dispositivi denominate che desideri sincronizzare. È necessario specificare i nomi esatti delle ombre.

`rateLimits`

(Facoltativo) Le impostazioni che determinano i limiti di velocità per le richieste di servizi shadow.

Questo oggetto contiene le seguenti informazioni.

`maxOutboundSyncUpdatesPerSecond`

(Facoltativo) Il numero massimo di richieste di sincronizzazione al secondo trasmesse dal dispositivo.

Impostazione predefinita: 100 richieste/secondo

`maxTotalLocalRequestsRate`

(Facoltativo) Il numero massimo di richieste IPC locali al secondo inviate al dispositivo principale.

Impostazione predefinita: 200 richieste/secondo

`maxLocalRequestsPerSecondPerThing`

(Facoltativo) Il numero massimo di richieste IPC locali al secondo inviate per ogni oggetto IoT connesso.

Impostazione predefinita: 20 richieste/secondo per ogni cosa

> **Note**
>
> Questi parametri relativi ai limiti di velocità definiscono il numero massimo di richieste al secondo per il servizio shadow locale. Il numero massimo di richieste al secondo per il servizio AWS IoT Device Shadow dipende dal tuo Regione AWS. Per ulteriori informazioni, consulta i limiti per l'API AWS IoT Device Shadow Service in Riferimenti generali di Amazon Web Services.

`shadowDocumentSizeLimitBytes`

(Facoltativo) La dimensione massima consentita di ogni documento di stato JSON per le ombre locali.

Se si aumenta questo valore, è necessario aumentare anche il limite di risorse per il documento di stato JSON per le ombre nel cloud. Per ulteriori informazioni, consulta i limiti per l'API AWS IoT Device Shadow Service in Riferimenti generali di Amazon Web Services.

Impostazione predefinita: 8192 byte

Massimo: 30720 byte

Example Esempio: aggiornamento di fusione della configurazione

L'esempio seguente mostra un esempio di aggiornamento di fusione della configurazione con tutti i parametri di configurazione disponibili per il componente shadow manager.

```
{
  "synchronize": {
    "coreThing": {
      "classic": true,
      "namedShadows": [
```

```
        "MyCoreShadowA",
        "MyCoreShadowB"
      ]
    },
    "shadowDocuments": [
      {
        "thingName": "MyDevice1",
        "classic": false,
        "namedShadows": [
          "MyShadowA",
          "MyShadowB"
        ]
      },
      {
        "thingName": "MyDevice2",
        "classic": true,
        "namedShadows": []
      }
    ]
  },
  "rateLimits": {
    "maxOutboundSyncUpdatesPerSecond": 100,
    "maxTotalLocalRequestsRate": 200,
    "maxLocalRequestsPerSecondPerThing": 20
  },
  "shadowDocumentSizeLimitBytes": 8192
}
```

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
2.3.7	Correzioni di bug e miglioramenti • Risolve un problema per cui lo shadow manager registra periodicamente un `NullPointerException` errore durante la sincronizzazione di uno shadow manager.
2.3.6	Correzioni di bug e miglioramenti • Risolve un problema per cui le proprietà shadow che vengono eliminate tramite Cloud AWS aggiornamenti mentre il dispositivo è offline continuano a esistere nell'ombra locale dopo il ripristino della connettiv ità.
2.3.5	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.3.4	Correzioni di bug e miglioramenti • Aggiunge il supporto per i documenti dello stato ombra nulli e vuoti.
2.3.3	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.

Versione	Modifiche
2.3.2	Correzioni di bug e miglioramenti • Risolve un problema per cui lo shadow manager entra nello `BROKEN` stato quando il database shadow locale è danneggiato. • Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.3.1	Correzioni di bug e miglioramenti • Risolve una condizione che potrebbe impedire la sincronizzazione degli aggiornamenti di Cloud Shadow. • Risolve un problema per cui le modifiche alla configurazione di Named Shadow Sync si applicano a una sola shadow denominata.
2.3.0	Correzioni di bug e miglioramenti • Risolve un problema che poteva impedire la sincronizzazione delle ombre quando la chiave privata del dispositivo Greengrass viene archiviata in un modulo di sicurezza hardware.
2.2.4	Correzioni di bug e miglioramenti • Risolve un problema per cui la convalida della dimensione dell'ombr a non era coerente con quella del cloud durante l'aggiornamento del documento shadow locale. • Risolve un problema per cui lo shadow manager interrompe l'ascolto degli aggiornamenti di configurazione se una distribuzione esegue un errore `RESET` sui nodi di configurazione.
2.2.3	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.2.2	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.2.1	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.

Versione	Modifiche
2.2.0	Nuove funzionalità • Aggiunge il supporto per il servizio shadow locale tramite l'interfaccia locale di pubblicazione/sottoscrizione. È ora possibile comunicare con il broker di messaggi di pubblicazione/sottoscrizione locale su argomenti Shadow MQTT per ottenere, aggiornare ed eliminare le ombre sul dispositivo principale. Questa funzionalità consente di connettere i dispositivi client al servizio shadow locale utilizzando il bridge MQTT per inoltrare messaggi su argomenti shadow tra i dispositivi client e l'interfa ccia locale di pubblicazione/sottoscrizione. Questa funzionalità richiede la versione 2.6.0 o successiva del componente Greengrass nucleus. Per connettere i dispositivi client al servizio shadow locale, è necessario utilizzare anche la versione 2.2.0 o successiva del componente bridge MQTT. • Aggiunge l'`direction` opzione che è possibile configurare per personalizzare la direzione di sincronizzazione delle ombre tra il servizio shadow locale e il. Cloud AWSÈ possibile configurare questa opzione per ridurre la larghezza di banda e le connessioni a. Cloud AWS
2.1.1	Correzioni di bug e miglioramenti • Risolve un problema per cui la profondità massima nelle `reported` sezioni `desired` e nelle sezioni del documento sullo stato ombra del dispositivo JSON era di 4 livelli anziché 5 livelli. • Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.0	Nuove funzionalità • Aggiunge il supporto per intervalli periodici di sincronizzazione shadow, in modo da poter configurare il dispositivo principale per ridurre l'utilizzo della larghezza di banda e i costi.
2.0.6	Questa versione contiene correzioni di bug e miglioramenti.
2.0.5	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.

Versione	Modifiche
2.0.4	Correzioni di bug e miglioramenti • Risolve un problema che causava l'eliminazione di versioni appena create di qualsiasi shadow precedentemente eliminata da Shadow Manager. • Aggiorna l'operazione `DeleteThingShadow` IPC per incrementare la versione shadow quando viene chiamata.
2.0.3	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.2	Correzioni di bug e miglioramenti • È stato risolto un problema che impediva al gestore di ombra di riconoscere la `delta` proprietà durante la sincronizzazione degli stati ombra da. AWS IoT Core • È stato risolto un problema che a volte causava l'unione errata delle richieste di sincronizzazione per un'ombra.
2.0.1	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.0.0	Versione iniziale.

Amazon SNS

Il componente Amazon SNS (`aws.greengrass.SNS`) pubblica messaggi su un argomento di Amazon Simple Notification Service (Amazon SNS). È possibile utilizzare questo componente per inviare eventi dai dispositivi core Greengrass a server Web, indirizzi e-mail e altri abbonati ai messaggi. Per ulteriori informazioni, consulta Cos'è Amazon SNS? nella Guida per gli sviluppatori di Amazon Simple Notification Service.

Per pubblicare su un argomento di Amazon SNS con questo componente, pubblica un messaggio sull'argomento a cui questo componente è abbonato. Per impostazione predefinita, questo componente sottoscrive l'argomento di pubblicazione/sottoscrizione `sns/message` locale. È possibile specificare altri argomenti, inclusi gli argomenti AWS IoT Core MQTT, quando si distribuisce questo componente.

Nel componente personalizzato, potresti voler implementare la logica di filtraggio o formattazione per elaborare i messaggi da altre fonti prima di pubblicarli su questo componente. Ciò consente di centralizzare la logica di elaborazione dei messaggi su un singolo componente.

> **Note**
>
> Questo componente offre funzionalità simili al connettore Amazon SNS nella AWS IoT Greengrass versione 1. Per ulteriori informazioni, consulta Amazon SNS Connector nella AWS IoT Greengrass V1 Developer Guide.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x
- 2,0x

Type

Questo componente è un componente Lambda () `aws.greengrass.lambda`. Il nucleo Greengrass esegue la funzione Lambda di questo componente utilizzando il componente di avvio Lambda.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato solo sui dispositivi principali di Linux.

Requisiti

Questo componente ha i seguenti requisiti:

- Il dispositivo principale deve soddisfare i requisiti per eseguire le funzioni Lambda. Se desideri che il dispositivo principale esegua funzioni Lambda containerizzate, il dispositivo deve soddisfare i requisiti per farlo. Per ulteriori informazioni, consulta Requisiti della funzione Lambda.
- Python versione 3.7 installata sul dispositivo principale e aggiunta alla variabile di ambiente PATH.
- Argomento Amazon SNS Per le istruzioni, consulta Creazione di un argomento Amazon SNS nella Guida per lo Sviluppatore di Amazon Simple Notification Service.
- Il ruolo del dispositivo Greengrass deve consentire l'`sns:Publish`azione, come illustrato nel seguente esempio di policy IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "sns:Publish"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:sns:region:account-id:topic-name"
      ]
    }
  ]
}
```

È possibile sovrascrivere dinamicamente l'argomento predefinito nel payload del messaggio di input per questo componente. Se l'applicazione utilizza questa funzionalità, la policy IAM deve includere tutti gli argomenti di destinazione come risorse. Puoi concedere alle risorse un accesso granulare o condizionale (ad esempio, utilizzando uno schema di denominazione con il carattere jolly *).

- Per ricevere i dati di output da questo componente, è necessario unire il seguente aggiornamento di configurazione per il componente legacy del router di abbonamento (`aws.greengrass.LegacySubscriptionRouter`) quando si distribuisce questo componente. Questa configurazione specifica l'argomento in cui questo componente pubblica le risposte.

  Legacy subscription router v2.1.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-sns": {
        "id": "aws-greengrass-sns",
        "source": "component:aws.greengrass.SNS",
        "subject": "sns/message/status",
        "target": "cloud"
      }
    }
  }
  ```

  Legacy subscription router v2.0.x

  ```
  {
    "subscriptions": {
      "aws-greengrass-sns": {
        "id": "aws-greengrass-sns",
        "source": "arn:aws:lambda:region:aws:function:aws-greengrass-sns:version",
        "subject": "sns/message/status",
        "target": "cloud"
      }
    }
  }
  ```

  - Sostituisci la *`regione`* con quella Regione AWS che usi.
  - Sostituisci la *`versione`* con la versione della funzione Lambda eseguita da questo componente. Per trovare la versione della funzione Lambda, è necessario visualizzare la ricetta per la versione di questo componente che si desidera distribuire. Apri la pagina dei dettagli di questo componente nella AWS IoT Greengrass console e cerca la coppia chiave-valore della funzione Lambda. Questa coppia chiave-valore contiene il nome e la versione della funzione Lambda.

> **Important**
>
> È necessario aggiornare la versione della funzione Lambda sul router di sottoscrizione legacy ogni volta che si distribuisce questo componente. Ciò garantisce l'utilizzo della versione corretta della funzione Lambda per la versione del componente che si distribuisce.

Per ulteriori informazioni, consulta Creare distribuzione.

- Il componente Amazon SNS è supportato per l'esecuzione in un VPC. Per distribuire questo componente in un VPC, è necessario quanto segue.
  - Il componente Amazon SNS deve disporre di una connettività con `sns.region.amazonaws.com` l'endpoint VPC di. `com.amazonaws.us-east-1.sns`

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`sns.region.amazonaws.com`	443	Sì	Pubblicare i messaggi su Amazon SNS.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le

dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.7

La tabella seguente elenca le dipendenze per la versione 2.1.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.6

La tabella seguente elenca le dipendenze per la versione 2.1.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.5

La tabella seguente elenca le dipendenze per la versione 2.1.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Rigidi

Dipendenza	Versioni compatibili	Tipo di dipendenza
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.4

La tabella seguente elenca le dipendenze per la versione 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.3

La tabella seguente elenca le dipendenze per la versione 2.1.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.2

La tabella seguente elenca le dipendenze per la versione 2.1.2 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.8 - 2.1.0

La tabella seguente elenca le dipendenze per le versioni 2.0.8 e 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.7

La tabella seguente elenca le dipendenze per la versione 2.0.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.6

La tabella seguente elenca le dipendenze per la versione 2.0.6 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.5

La tabella seguente elenca le dipendenze per la versione 2.0.5 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili

Dipendenza	Versioni compatibili	Tipo di dipendenza
Servizio di scambio di token	^2.0.0	Rigidi

2.0.4

La tabella seguente elenca le dipendenze per la versione 2.0.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Rigidi
Lanciatore Lambda	^2.0.0	Rigidi
Runtime Lambda	^2,0,0	Flessibili
Servizio di scambio di token	^2.0.0	Rigidi

2.0.3

La tabella seguente elenca le dipendenze per la versione 2.0.3 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Rigidi
Lanciatore Lambda	>=1.0.0	Rigidi
Runtime Lambda	>=1.0.0	Flessibili
Servizio di scambio di token	>=1.0.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

> **Note**
>
> La configurazione predefinita di questo componente include i parametri della funzione Lambda. Ti consigliamo di modificare solo i seguenti parametri per configurare questo componente sui tuoi dispositivi.

`lambdaParams`

Un oggetto che contiene i parametri per la funzione Lambda di questo componente. Questo oggetto contiene le seguenti informazioni:

`EnvironmentVariables`

Un oggetto che contiene i parametri della funzione Lambda. Questo oggetto contiene le seguenti informazioni:

`DEFAULT_SNS_ARN`

L'ARN dell'argomento predefinito di Amazon SNS in cui questo componente pubblica i messaggi. Puoi sovrascrivere l'argomento di destinazione con la `sns_topic_arn` proprietà nel payload del messaggio di input.

`containerMode`

(Facoltativo) La modalità di containerizzazione per questo componente. Seleziona una delle opzioni seguenti:

- `NoContainer`— Il componente non viene eseguito in un ambiente di runtime isolato.
- `GreengrassContainer`— Il componente viene eseguito in un ambiente di runtime isolato all'interno del AWS IoT Greengrass contenitore.

Impostazione predefinita: `GreengrassContainer`

`containerParams`

(Facoltativo) Un oggetto che contiene i parametri del contenitore per questo componente. Il componente utilizza questi parametri se si specifica `GreengrassContainer` per`containerMode`.

Questo oggetto contiene le seguenti informazioni:

`memorySize`

(Facoltativo) La quantità di memoria (in kilobyte) da allocare al componente.

Il valore predefinito è 512 MB (525.312 KB).

`pubsubTopics`

(Facoltativo) Un oggetto che contiene gli argomenti a cui il componente sottoscrive la sottoscrizione per ricevere messaggi. È possibile specificare ogni argomento e se il componente sottoscrive gli argomenti MQTT AWS IoT Core o gli argomenti di pubblicazione/sottoscrizione locali.

Questo oggetto contiene le seguenti informazioni:

`0`— Si tratta di un indice di matrice sotto forma di stringa.

Un oggetto che contiene le seguenti informazioni:

`type`

(Facoltativo) Il tipo di messaggi di pubblicazione/sottoscrizione utilizzato da questo componente per sottoscrivere i messaggi. Seleziona una delle opzioni seguenti:

- `PUB_SUB`: iscriviti ai messaggi di pubblicazione/sottoscrizione locali. Se scegli questa opzione, l'argomento non può contenere caratteri jolly MQTT. Per ulteriori informazioni su come inviare messaggi dal componente personalizzato quando si specifica questa opzione, vedere. Pubblicare/sottoscrivere messaggi locali
- `IOT_CORE`— Abbonarsi ai messaggi AWS IoT Core MQTT. Se scegli questa opzione, l'argomento può contenere caratteri jolly MQTT. Per ulteriori informazioni su come inviare messaggi da componenti personalizzati quando si specifica questa opzione, vedere. AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

Impostazione predefinita: `PUB_SUB`

`topic`

(Facoltativo) L'argomento a cui il componente si iscrive per ricevere messaggi. Se si specifica `IotCore` for`type`, è possibile utilizzare i caratteri jolly MQTT (+and#) in questo argomento.

Example Esempio: aggiornamento basato sull'unione della configurazione (modalità contenitore)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_SNS_ARN": "arn:aws:sns:us-west-2:123456789012:mytopic"
    }
  },
  "containerMode": "GreengrassContainer"
}
```

Example Esempio: aggiornamento tramite fusione della configurazione (nessuna modalità contenitore)

```
{
  "lambdaExecutionParameters": {
    "EnvironmentVariables": {
      "DEFAULT_SNS_ARN": "arn:aws:sns:us-west-2:123456789012:mytopic"
    }
  },
  "containerMode": "NoContainer"
}
```

Dati di input

Questo componente accetta messaggi sul seguente argomento e pubblica il messaggio così com'è nell'argomento Amazon SNS di destinazione. Per impostazione predefinita, questo componente sottoscrive i messaggi di pubblicazione/sottoscrizione locali. Per ulteriori informazioni su come pubblicare messaggi su questo componente dai componenti personalizzati, consulta. Pubblicare/sottoscrivere messaggi locali

Argomento predefinito (pubblicazione/sottoscrizione locale): `sns/message`

Il messaggio accetta le seguenti proprietà. I messaggi di input devono essere in formato JSON.

`request`

Le informazioni sul messaggio da inviare all'argomento Amazon SNS.

Tipo: `object` che contiene le seguenti informazioni:

`message`

Il contenuto del messaggio sotto forma di stringa.

Per inviare un oggetto JSON, serializzatelo come stringa e specificate `json` la `message_structure` proprietà.

Tipo: `string`

`subject`

(Facoltativo) L'oggetto del messaggio.

Tipo: `string`

L'oggetto può essere testo ASCII e contenere fino a 100 caratteri. Deve iniziare con una lettera, un numero o un segno di punteggiatura. Non può includere interruzioni di riga o caratteri di controllo.

`sns_topic_arn`

(Facoltativo) L'ARN dell'argomento Amazon SNS in cui questo componente pubblica il messaggio. Specificare questa proprietà per sovrascrivere l'argomento predefinito di Amazon SNS.

Tipo: `string`

`message_structure`

(Facoltativo) La struttura del messaggio. Specificate di `json` inviare un messaggio JSON da serializzare come stringa nella `content` proprietà.

Tipo: `string`

Valori validi: `json`

`id`

Un ID arbitrario della richiesta. Utilizzate questa proprietà per mappare una richiesta di input a una risposta di output. Quando specificate questa proprietà, il componente imposta la `id` proprietà nell'oggetto di risposta su questo valore.

Tipo: `string`

> **Note**
>
> La dimensione del messaggio può essere al massimo di 256 KB.

Example Input di esempio: messaggio in formato stringa

```
{
  "request": {
    "subject": "Message subject",
    "message": "Message data",
    "sns_topic_arn": "arn:aws:sns:region:account-id:topic2-name"
  },
  "id": "request123"
}
```

Example Input di esempio: messaggio JSON

```
{
  "request": {
    "subject": "Message subject",
    "message": "{ \"default\": \"Message data\" }",
    "message_structure": "json"
  },
  "id": "request123"
}
```

Dati di output

Per impostazione predefinita, questo componente pubblica le risposte come dati di output sul seguente argomento MQTT. È necessario specificare questo argomento come contenuto `subject` nella configurazione del componente legacy del router di abbonamento. Per ulteriori informazioni su come sottoscrivere i messaggi relativi a questo argomento nei componenti personalizzati, consultaAWS IoT Core Pubblicare/sottoscrivere messaggi MQTT.

Argomento predefinito (AWS IoT Core MQTT): `sns/message/status`

Example Output di esempio: Operazione riuscita

```
{
```

```
  "response": {
    "sns_message_id": "f80a81bc-f44c-56f2-a0f0-d5af6a727c8a",
    "status": "success"
  },
  "id": "request123"
}
```

Example Esempio di output: Errore

```
{
  "response" : {
    "error": "InvalidInputException",
    "error_message": "SNS Topic Arn is invalid",
    "status": "fail"
  },
  "id": "request123"
}
```

File di registro locale

Questo componente utilizza il seguente file di registro.

```
/greengrass/v2/logs/aws.greengrass.SNS.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale.

  ```
  sudo tail -f /greengrass/v2/logs/aws.greengrass.SNS.log
  ```

Licenze

Questo componente include i seguenti software/licenze di terze parti:

- AWS SDK for Python (Boto3)/Apache License 2.0
- botocore/Apache License 2.0
- dateutil/PSF License

- docutils/BSD License, GNU General Public License (GPL), Python Software Foundation License, Public Domain
- jmespath/MIT License
- s3transfer/Apache License 2.0
- urllib3/MIT License

Questo componente è rilasciato in base al contratto di licenza del software Greengrass Core.

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
2.1.7	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.1.6	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.5	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.4	Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.3	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.
2.1.2	Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.1.1	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.1.0	Nuove funzionalità • Aggiunge il supporto per le configurazioni proxy di rete HTTPS. Per ulteriori informazioni, consultare Connessione alla porta 443 o tramite un proxy di rete e Abilita il dispositivo principale in modo che consideri attendibile un proxy HTTPS.
2.0.8	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.0.7	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.6	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.

Versione	Modifiche
2.0.5	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.0.4	Versione aggiornata per la versione 2.1.0 di Greengrass nucleus.
2.0.3	Versione iniziale.

Stream manager

Il componente stream manager (`aws.greengrass.StreamManager`) consente di elaborare flussi di dati da trasferire Cloud AWS dai dispositivi core Greengrass.

Per ulteriori informazioni su come configurare e utilizzare stream manager nei componenti personalizzati, vedere. Gestisci i flussi di dati sui dispositivi core Greengrass

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.1.x
- 2,0x

> **Note**
>
> Se utilizzi stream manager per esportare dati nel cloud, non puoi aggiornare la versione 2.0.7 del componente stream manager a una versione compresa tra v2.0.8 e v2.0.11. Se stai implementando stream manager per la prima volta, ti consigliamo vivamente di distribuire la versione più recente del componente stream manager.

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Il ruolo di scambio di token deve consentire l'accesso alle Cloud AWS destinazioni utilizzate con stream manager. Per ulteriori informazioni, consultare:
  - the section called “Canali AWS IoT Analytics”
  - the section called “Flussi di dati Amazon Kinesis”
  - the section called “AWS IoT SiteWiseproprietà degli asset”
  - the section called “Oggetti Amazon S3”
- Il componente stream manager è supportato per l'esecuzione in un VPC. Per distribuire questo componente in un VPC, è necessario quanto segue.
  - Il componente stream manager deve essere connesso al AWS servizio su cui si pubblicano i dati.

  - Amazon S3: `com.amazonaws.region.s3`
  - Amazon Kinesis Data Streams: `com.amazonaws.region.kinesis-streams`
  - AWS IoT SiteWise: `com.amazonaws.region.iotsitewise.data`
- Se pubblichi dati su Amazon S3 nella `us-east-1` regione, questo componente tenterà di utilizzare l'endpoint globale S3 per impostazione predefinita; tuttavia, questo endpoint non è disponibile tramite l'endpoint dell'interfaccia VPC di Amazon S3. Per ulteriori informazioni, consulta Restrizioni e limitazioni AWS PrivateLink di Amazon S3. Per risolvere questo problema, puoi scegliere tra le seguenti opzioni.
  - Configura il componente stream manager per utilizzare l'endpoint S3 regionale nella `us-east-1` regione, fornendo la variabile di `AWS_S3_US_EAST_1_REGIONAL_ENDPOINT=regional` ambiente.
  - Crea un endpoint VPC gateway Amazon S3 anziché un endpoint VPC con interfaccia Amazon S3. Gli endpoint gateway S3 supportano l'accesso all'endpoint globale S3. Per ulteriori informazioni, consulta Creare un endpoint gateway.

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`iotanalytics. region.amazonaws.com`	443	No	Obbligatorio se si pubblicano dati su. AWS IoT Analytics
`kinesis.region.amazonaws.com`	443	No	Obbligatorio se si pubblican

Endpoint	Porta	Richiesto	Descrizione
			o dati su Firehose.
`data.iotsitewise. region.amazonaws.com`	443	No	Obbligatorio se si pubblicano dati su. AWS IoT SiteWise
`*.s3.amazonaws.com`	443	No	Obbligatorio se pubblichi dati su bucket S3. Puoi sostituire lo * con il nome di ogni bucket in cui pubblichi i dati.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

2.1.11

La tabella seguente elenca le dipendenze per le versioni da 2.1.11 a 2.1.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.13.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.1.9 – 2.1.10

La tabella seguente elenca le dipendenze per le versioni da 2.1.9 a 2.1.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.12.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.1.5 – 2.1.8

La tabella seguente elenca le dipendenze per le versioni da 2.1.5 a 2.1.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.11.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.1.2 – 2.1.4

La tabella seguente elenca le dipendenze per le versioni da 2.1.2 a 2.1.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.10.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.1.1

La tabella seguente elenca le dipendenze per la versione 2.1.1 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.9.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.1.0

La tabella seguente elenca le dipendenze per la versione 2.1.0 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.8.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.0.15

La tabella seguente elenca le dipendenze per la versione 2.0.15 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.7.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.0.13 and 2.0.14

La tabella seguente elenca le dipendenze per le versioni 2.0.13 e 2.0.14 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.6.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.0.11 and 2.0.12

La tabella seguente elenca le dipendenze per le versioni 2.0.11 e 2.0.12 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.5.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.0.10

La tabella seguente elenca le dipendenze per la versione 2.0.10 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.4.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.0.9

La tabella seguente elenca le dipendenze per la versione 2.0.9 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.3.0	Flessibili

Dipendenza	Versioni compatibili	Tipo di dipendenza
Servizio di scambio di token	>=0.0.0	Rigidi

2.0.8

La tabella seguente elenca le dipendenze per la versione 2.0.8 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.0 <2.2.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

2.0.7

La tabella seguente elenca le dipendenze per la versione 2.0.7 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.0.3 <2.1.0	Flessibili
Servizio di scambio di token	>=0.0.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`STREAM_MANAGER_STORE_ROOT_DIR`

(Facoltativo) Il percorso assoluto della directory locale utilizzata per archiviare gli stream. Questo valore deve iniziare con una barra (ad esempio, `/data`).

È necessario specificare una cartella esistente e l'utente di sistema che esegue il componente stream manager deve disporre delle autorizzazioni per leggere e scrivere in questa cartella. Ad esempio, è possibile eseguire i seguenti comandi per creare e configurare una cartella `/var/greengrass/streams`, specificata come cartella principale dello stream manager. Questi comandi consentono all'utente di sistema predefinito di leggere e scrivere in questa cartella. `ggc_user`

```
sudo mkdir /var/greengrass/streams
sudo chown ggc_user /var/greengrass/streams
sudo chmod 700 /var/greengrass/streams
```

Impostazione predefinita: */greengrass/v2*`/work/aws.greengrass.StreamManager`

`STREAM_MANAGER_SERVER_PORT`

(Facoltativo) Il numero di porta locale da utilizzare per comunicare con lo stream manager.

È possibile specificare `0` di utilizzare una porta disponibile in modo casuale.

Impostazione predefinita: `8088`

`STREAM_MANAGER_AUTHENTICATE_CLIENT`

(Facoltativo) Puoi rendere obbligatoria l'autenticazione dei client prima che possano interagire con lo stream manager. L'SDK Stream Manager controlla l'interazione tra i client e lo stream manager. Questo parametro determina quali client possono chiamare l'SDK Stream Manager per lavorare con gli stream. Per ulteriori informazioni, consulta Stream Manager Client Authentication.

Se lo specifichi`true`, Stream Manager SDK consente solo i componenti Greengrass come client.

Se lo specifichi`false`, Stream Manager SDK consente a tutti i processi sul dispositivo principale di essere client.

Impostazione predefinita: `true`

`STREAM_MANAGER_EXPORTER_MAX_BANDWIDTH`

(Facoltativo) La larghezza di banda massima media (in kilobit al secondo) che lo stream manager può utilizzare per esportare i dati.

Impostazione predefinita: nessun limite

`STREAM_MANAGER_EXPORTER_THREAD_POOL_SIZE`

(Facoltativo) Il numero massimo di thread attivi che lo stream manager può utilizzare per esportare i dati.

La dimensione ottimale dipende dall'hardware, dal volume del flusso e dal numero pianificato di flussi di esportazione. Se la velocità di esportazione è bassa, puoi regolare questa impostazione per trovare la dimensione ottimale per l'hardware e il business case. La CPU e la memoria dell'hardware del dispositivo core sono fattori limitanti. Per iniziare, è possibile provare a impostare questo valore uguale al numero di core di processore sul dispositivo.

Fare attenzione a non impostare una dimensione superiore a quella supportata dall'hardware. Ogni stream consuma risorse hardware, quindi cercate di limitare il numero di flussi di esportazione su dispositivi con restrizioni.

Impostazione predefinita: 5 thread

`STREAM_MANAGER_EXPORTER_S3_DESTINATION_MULTIPART_UPLOAD_MIN_PART_SIZE_BYTES`

(Facoltativo) La dimensione minima (in byte) di una parte in un caricamento multiparte su Amazon S3. Stream Manager utilizza questa impostazione e la dimensione del file di input per determinare come raggruppare i dati in una richiesta PUT composta da più parti.

> **Note**
>
> Stream Manager utilizza la `sizeThresholdForMultipartUploadBytes` proprietà streams per determinare se esportare in Amazon S3 come caricamento singolo o multiparte. AWS IoT Greengrass i componenti possono impostare questa soglia quando creano uno stream che esporta in Amazon S3.

Impostazione predefinita: `5242880` (5 MB). Questo è anche il valore minimo.

`LOG_LEVEL`

(Facoltativo) Il livello di registrazione per il componente. Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `TRACE`
- `DEBUG`
- `INFO`

- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

`JVM_ARGS`

(Facoltativo) Gli argomenti personalizzati della Java Virtual Machine da passare allo stream manager all'avvio. Separa più argomenti per spazi.

Utilizza questo parametro solo quando devi sostituire le impostazioni predefinite utilizzate dalla JVM. Ad esempio, potrebbe essere necessario aumentare la dimensione heap predefinita se prevedi di esportare un numero elevato di flussi.

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica l'utilizzo di una porta non predefinita.

```
{
  "STREAM_MANAGER_SERVER_PORT": "18088"
}
```

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.greengrass.StreamManager.log
```

Windows

```
C:\greengrass\v2\logs\aws.greengrass.StreamManager.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

Linux

```
sudo tail -f /greengrass/v2/logs/aws.greengrass.StreamManager.log
```

Windows (PowerShell)

```
Get-Content C:\greengrass\v2\logs\aws.greengrass.StreamManager.log -Tail 10 -
Wait
```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.1.12	Correzioni di bug e miglioramenti Aggiorna l'ordine in cui vengono utilizzate le credenziali in modo che le credenziali Greengrass siano preferite AWS per le richieste di assistenza.
2.1.11	Versione aggiornata per la versione 2.12.0 di Greengrass nucleus.
2.1.10	Correzioni di bug e miglioramenti Risolve un problema per cui la configurazione del proxy HTTPS non considera attendibile la catena di certificati dell'autorità di certificazione (CA) Greengrass.
2.1.9	Versione aggiornata per la versione 2.11.0 di Greengrass nucleus.
2.1.8	Correzioni di bug e miglioramenti Risolve un problema per cui lo stream manager SiteWise riprova all'infinito le esportazioni con esito negativo. `InvalidRequestException`

Versione	Modifiche
2.1.7	Correzioni di bug e miglioramenti Risolve un problema per cui lo stream manager non riesce a leggere correttamente la configurazione del proxy.
2.1.6	Correzioni di bug e miglioramenti Risolve un problema che poteva causare un arresto anomalo all'avvio su alcuni processori ARMv8, incluso Jetson Nano.
2.1.5	Versione aggiornata per la versione 2.10.0 di Greengrass nucleus.
2.1.4	Correzioni di bug e miglioramenti • Risolve un problema per cui le voci relative alla stessa risorsa di proprietà con lo stesso timestamp all'interno di un singolo batch restituiv ano `ConflictingOperationException` dall' SiteWise API, il che faceva sì che lo stream manager riprovasse continuamente. • Aggiorna il timeout di connessione predefinito da 3 secondi a 1 minuto.
2.1.3	Correzioni di bug e miglioramenti Risolve un problema di avvio nel sistema operativo Windows quando viene eseguito come utente SYSTEM.
2.1.2	Correzioni di bug e miglioramenti • Risolve un problema nel sistema operativo Windows che utilizza una lingua diversa dall'inglese. • Versione aggiornata per la versione 2.9.0 di Greengrass nucleus.
2.1.1	Versione aggiornata per la versione 2.8.0 di Greengrass nucleus.

Versione	Modifiche
2.1.0	Nuove funzionalità • Aggiorna questo componente per inviare automaticamente i parametri di telemetria ad Amazon. EventBridge Per ulteriori informazioni, consulta Raccogli dati di telemetria sanitaria del sistema dai dispositivi principali AWS IoT Greengrass. Questa funzionalità richiede la versione 2.7.0 o successiva del componente Greengrass nucleus. • Versione aggiornata per la versione 2.7.0 di Greengrass nucleus.
2.0.15	Versione aggiornata per la versione 2.6.0 di Greengrass nucleus.
2.0.14	Questa versione contiene correzioni di bug e miglioramenti.
2.0.13	Versione aggiornata per la versione 2.5.0 di Greengrass nucleus.
2.0.12	Correzioni di bug e miglioramenti Risolve un problema che impediva l'aggiornamento dello stream manager v2.0.7 a una versione compresa tra v2.0.8 e v2.0.11. Se utilizzi stream manager per esportare dati nel cloud, ora puoi eseguire l'aggiornamento alla v2.0.12.
2.0.11	Versione aggiornata per la versione 2.4.0 di Greengrass nucleus.
2.0.10	Versione aggiornata per la versione 2.3.0 di Greengrass nucleus.
2.0.9	Versione aggiornata per la versione 2.2.0 di Greengrass nucleus.
2.0.8	Versione aggiornata per la versione 2.1.0 di Greengrass nucleus.
2.0.7	Versione iniziale.

Agente Systems Manager

Il componente AWS Systems Manager Agent (`aws.greengrass.SystemsManagerAgent`) installa Systems Manager Agent, in modo da poter gestire i dispositivi principali con Systems

Manager. Systems Manager è un AWS servizio che puoi utilizzare per visualizzare e controllare la tua infrastruttura AWS, comprese le istanze Amazon EC2, i server e le macchine virtuali (VM) locali e i dispositivi edge. Systems Manager consente di visualizzare i dati operativi, automatizzare le attività operative e mantenere la sicurezza e la conformità. Per ulteriori informazioni, vedere Cos'è AWS Systems Manager? e Informazioni su Systems Manager Agent nella Guida AWS Systems Manager per l'utente.

Gli strumenti e le funzionalità di Systems Manager sono denominati funzionalità. I dispositivi core Greengrass supportano tutte le funzionalità di Systems Manager. Per ulteriori informazioni su queste funzionalità e su come utilizzare Systems Manager per gestire i dispositivi principali, vedere le funzionalità di Systems Manager nella Guida per l'AWS Systems Manager utente.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 1.1.x
- 1.0.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato solo sui dispositivi principali di Linux.

Requisiti

Questo componente ha i seguenti requisiti:

- Un dispositivo core Greengrass che funziona su una piattaforma Linux a 64 bit: Armv8 (AArch64) o x86_64.
- È necessario disporre di un ruolo di servizio AWS Identity and Access Management (IAM) che Systems Manager possa assumere. Questo ruolo deve includere la politica ManagedInstanceCore gestita di AmazonSSM o una politica personalizzata che definisca autorizzazioni equivalenti. Per ulteriori informazioni, consulta Creare un ruolo di servizio IAM per dispositivi periferici nella Guida per l'utente.AWS Systems Manager

  Quando distribuisci questo componente, devi specificare il nome di questo ruolo per il parametro di `SSMRegistrationRole` configurazione.
- Il ruolo del dispositivo Greengrass deve consentire le azioni `ssm:AddTagsToResource` `essm:RegisterManagedInstance`. Il ruolo del dispositivo deve inoltre consentire l'`iam:PassRole`azione del ruolo del servizio IAM che soddisfa il requisito precedente. Il seguente esempio di policy IAM concede queste autorizzazioni.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iam::account-id:role/SSMServiceRole"
      ]
    },
    {
      "Action": [
        "ssm:AddTagsToResource",
        "ssm:RegisterManagedInstance"
      ],
      "Effect": "Allow",
```

```
      "Resource": "*"
    }
  ]
}
```

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`ec2messages.region.amazonaws.com`	443	Sì	Comunica con il servizio Systems Manager in Cloud AWS.
`ssm.region.amazonaws.com`	443	Sì	Registrare il dispositivo principale come nodo gestito da Systems Manager.
`ssmmessages.region.amazonaws.com`	443	Sì	Comunica con Session Manager, una funzionalità di Systems

Endpoint	Porta	Richiesto	Descrizione
			Manager, in Cloud AWS.

Per ulteriori informazioni, consulta Riferimento: ec2messages, ssmmessages e altre chiamate API nella Guida per l'utente.AWS Systems Manager

Dipendenze

Quando distribuisci un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle sue dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

La tabella seguente elenca le dipendenze per le versioni da 1.0.0 a 1.2.4 di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Servizio di scambio di token	^2.0.0	Flessibili

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente fornisce i seguenti parametri di configurazione che è possibile personalizzare durante la distribuzione del componente.

`SSMRegistrationRole`

Il ruolo del servizio IAM che Systems Manager può assumere e che include la policy ManagedInstanceCore gestita di AmazonSSM o una policy personalizzata che definisce

autorizzazioni equivalenti. Per ulteriori informazioni, consulta Creare un ruolo di servizio IAM per dispositivi periferici nella Guida per l'utente.AWS Systems Manager

`SSMOverrideExistingRegistration`

(Facoltativo) Se il dispositivo principale esegue già il Systems Manager Agent registrato con un'attivazione ibrida, è possibile sovrascrivere la registrazione Systems Manager Agent esistente del dispositivo. Imposta questa opzione `true` per registrare il dispositivo principale come nodo gestito utilizzando l'agente Systems Manager fornito da questo componente.

> **Note**
>
> Questa opzione si applica solo ai dispositivi registrati con un'attivazione ibrida. Se il dispositivo principale viene eseguito su un'istanza Amazon EC2 con Systems Manager Agent installato e un ruolo del profilo dell'istanza configurato, l'ID del nodo gestito esistente dell'istanza Amazon EC2 inizia con. `i-` Quando si installa il componente Systems Manager Agent, l'agente Systems Manager registra un nuovo nodo gestito il cui ID inizia con `mi-` invece di`i-`. Quindi, è possibile utilizzare il nodo gestito il cui ID inizia con `mi-` per gestire il dispositivo principale con Systems Manager.

Impostazione predefinita: `false`

`SSMResourceTags`

(Facoltativo) I tag da aggiungere al nodo gestito di Systems Manager che questo componente crea per il dispositivo principale. È possibile utilizzare questi tag per gestire gruppi di dispositivi principali con Systems Manager. Ad esempio, è possibile eseguire un comando su tutti i dispositivi che dispongono di un tag specificato dall'utente.

Specificate un elenco in cui ogni tag è un oggetto con un `Key` e un`Value`. Ad esempio, il seguente valore di `SSMResourceTags` indica a questo componente di impostare il **`Owner`** tag **`richard-roe`** sul nodo gestito del dispositivo principale.

```
[
  {
    "Key": "Owner",
    "Value": "richard-roe"
  }
]
```

Questo componente ignora questi tag se il nodo gestito esiste già e `SSMOverrideExistingRegistration` lo è. `false`

Example Esempio: fusione e aggiornamento della configurazione

La configurazione di esempio seguente specifica l'utilizzo di un ruolo di servizio denominato `SSMServiceRole` per consentire al dispositivo principale di registrarsi e comunicare con Systems Manager.

```
{
  "SSMRegistrationRole": "SSMServiceRole",
  "SSMOverrideExistingRegistration": false,
  "SSMResourceTags": [
    {
      "Key": "Owner",
      "Value": "richard-roe"
    },
    {
      "Key": "Team",
      "Value": "solar"
    }
  ]
}
```

File di registro locale

Il software Systems Manager Agent scrive i log in una cartella esterna alla cartella principale di Greengrass. Per ulteriori informazioni, vedere Visualizzazione dei log di Systems Manager Agent nella Guida per l'AWS Systems Manager utente.

Il componente Systems Manager Agent utilizza script di shell per installare, avviare e arrestare Systems Manager Agent. È possibile trovare l'output di questi script nel seguente file di registro.

```
/greengrass/v2/logs/aws.greengrass.SystemsManagerAgent.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale.

```
sudo tail -f /greengrass/v2/logs/aws.greengrass.SystemsManagerAgent.log
```

Consulta anche

- Gestisci i dispositivi core Greengrass con AWS Systems Manager
- Cos'è AWS Systems Manager? nella Guida per l'utente di AWS Systems Manager
- Informazioni su Systems Manager Agent nella Guida AWS Systems Manager per l'utente

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
1.2.4	Correzioni di bug e miglioramenti Aggiorna questo componente per ottenere la versione 3.2.2303.0 dell'agente.
1.2.3	Correzioni di bug e miglioramenti • Aggiunge nuovi tentativi per l'installazione del componente Agent con snap su Greengrass. • Aggiorna la configurazione del componente Agent per utilizzare solo Onprem Identity in Greengrass. • Aggiorna questo componente per aggiornare l'agente solo quando la versione dell'agente installato non corrisponde alla versione del componente Greengrass SSM Agent.
1.1.0	Questa versione contiene correzioni di bug e miglioramenti.
1.0.0	Versione iniziale.

Servizio di scambio di token

Il componente del servizio di scambio di token (`aws.greengrass.TokenExchangeService`) fornisce AWS credenziali che è possibile utilizzare per interagire con AWS i servizi dei componenti personalizzati.

Il servizio di scambio di token esegue un'istanza di container Amazon Elastic Container Service (Amazon ECS) come server locale. Questo server locale si connette al provider di AWS IoT credenziali utilizzando l'alias del AWS IoT ruolo configurato nel componente core nucleus di Greengrass. Il componente fornisce due variabili di ambiente e. `AWS_CONTAINER_CREDENTIALS_FULL_URI` `AWS_CONTAINER_AUTHORIZATION_TOKEN` `AWS_CONTAINER_CREDENTIALS_FULL_URI`definisce l'URI di questo server locale. Quando un componente crea un client AWS SDK, il client riconosce questa variabile di ambiente URI e utilizza il token in `AWS_CONTAINER_AUTHORIZATION_TOKEN` per connettersi al servizio di scambio di token e recuperare AWS le credenziali. Ciò consente ai dispositivi core Greengrass di chiamare le operazioni di AWS assistenza. Per ulteriori informazioni su come utilizzare questo componente nei componenti personalizzati, vedereInteragisci con AWS i servizi.

> **Important**
>
> Il supporto per l'acquisizione di AWS credenziali in questo modo è stato aggiunto agli AWS SDK il 13 luglio 2016. Il componente deve utilizzare una versione AWS SDK creata a partire da tale data. Per ulteriori informazioni, consulta Using a support AWS SDK nella Amazon Elastic Container Service Developer Guide.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 2.0.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Dipendenze

Questo componente non ha alcuna dipendenza.

Configurazione

Questo componente non ha parametri di configurazione.

File di registro locale

Questo componente utilizza lo stesso file di registro del componente Greengrass nucleus.

Linux

```
/greengrass/v2/logs/greengrass.log
```

Windows

```
C:\greengrass\v2\logs\greengrass.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche in ogni versione del componente.

Versione	Modifiche
2.0.3	Versione iniziale.

Collettore IoT SiteWise OPC-UA

Il componente di raccolta IoT SiteWise OPC-UA (`aws.iot.SiteWiseEdgeCollectorOpcua`) consente ai AWS IoT SiteWise gateway di raccogliere dati dai server OPC-UA locali.

Con questo componente, i AWS IoT SiteWise gateway possono connettersi a più server OPC-UA. Per ulteriori informazioni sui AWS IoT SiteWise gateway, vedere Using AWS IoT SiteWise at the edge nella Guida per l'utente.AWS IoT SiteWise

Argomenti

- Versioni
- Type
- Sistema operativo
- Requisiti

- Dipendenze
- Configurazione
- Dati di input
- Dati di output
- File di registro locale
- Risoluzione dei problemi e debug
- Licenze
- Changelog
- Consulta anche

Versioni

Questo componente ha le seguenti versioni:

- 2.4.x
- 2.3.x
- 2.2.x
- 2.1.x
- 2,0x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Il dispositivo principale Greengrass deve funzionare su una delle seguenti piattaforme:
  - Sistema operativo: Ubuntu 18.04 o successivo

    Architettura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - Sistema operativo: Red Hat Enterprise Linux (RHEL) 8

    Architettura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - Sistema operativo: Amazon Linux 2

    Architettura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - Sistema operativo: Debian 11

    Architettura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - Sistema operativo: Windows Server 2019 o versione successiva

    Architettura: x86_64 (AMD64)
- Il dispositivo principale Greengrass deve consentire la connettività di rete in uscita ai server OPC-UA.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle sue dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

La tabella seguente elenca le dipendenze per tutte le versioni di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.3.0 <3.0.0	Rigidi

Dipendenza	Versioni compatibili	Tipo di dipendenza
Stream manager	>=2.3.0 2,0,10<3,0,0	Rigidi
Gestore segreto	>2,0,10 =2,0,8 <3,0,0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente non ha parametri di configurazione.

Puoi utilizzare la AWS IoT SiteWise console o l'API per configurare il componente del collettore IoT SiteWise OPC-UA. Per ulteriori informazioni, consulta Passaggio 4: Aggiungere fonti di dati, facoltativo nella Guida per l'AWS IoT SiteWise utente.

Dati di input

Questo componente accetta solo dati nei seguenti formati, tutti gli altri verranno ignorati e scartati. La tabella seguente associa i tipi di dati OPC UA ai loro equivalenti. SiteWise

SiteWise tipo di dati	tipo di dati OPC UA	Descrizione

`STRING`	`String`
	`Guid`
	`XmlElement`
		Una stringa di lunghezza massima di 1024 byte.

`INTEGER`	`SByte`
	`Byte`
	`Int16`
	`UInt16`
	`Int32`
	`UInt32*`
		Un numero intero con segno a 32 bit con un intervallo da. `-2,147,483,648 to 2,147,483,647`

SiteWise tipo di dati	tipo di dati OPC UA	Descrizione
	`Int64*`	

`DOUBLE`	`UInt32*`
	`Int64*`
	`Float`
	`Double`
		Un numero a virgola mobile con intervallo `-10^100 to 10^100` e `IEEE 754` precisione doppia.

`BOOLEAN`	`Boolean`	`true` o `false`.

* Per i tipi di dati OPC UA `UInt32` e`Int64`, il suo tipo di SiteWise dati sarà `INTEGER` se SiteWise è in grado di rappresentarne il valore, altrimenti lo sarà. `DOUBLE`

Dati di output

Questo componente scrive `BatchPutAssetPropertyValue` messaggi nello AWS IoT Greengrass stream manager. Per ulteriori informazioni, consulta BatchPutAssetPropertyValue nella documentazione di riferimento dell'API AWS IoT SiteWise .

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgeCollectorOpcua.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeCollectorOpcua.log -Tail
   10 -Wait
  ```

Risoluzione dei problemi e debug

Questo componente include un nuovo registro degli eventi per aiutare i clienti a identificare e risolvere i problemi. Il file di registro è separato dal file di registro locale e si trova nella seguente posizione. Sostituire */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgeCollectorOpcua/logs/
IotSiteWiseOpcUaCollectorEvents.log
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgeCollectorOpcua\logs
\IotSiteWiseOpcUaCollectorEvents.log
```

Questo registro include informazioni dettagliate e istruzioni per la risoluzione dei problemi. Oltre alla diagnostica, vengono fornite informazioni sulla risoluzione dei problemi, con una descrizione di come risolvere il problema e, a volte, con collegamenti a ulteriori informazioni. Le informazioni diagnostiche includono quanto segue:

- Livello di gravità

- Timestamp
- Informazioni aggiuntive specifiche sull'evento

Example Log di esempio

```
dataSourceConnectionSuccess:
  Summary: Successfully connected to OpcUa server
  Level: INFO
  Timestamp: '2023-06-15T21:04:16.303Z'
  Description: Successfully connected to the data source.
  AssociatedMetrics:
  - Name: FetchedDataStreams
    Description: The number of fetched data streams for this data source
    Value: 1.0
    Namespace: IoTSiteWise
    Dimensions:
    - Name: SourceName
      Value: SourceName{value=OPC-UA Server}
    - Name: ThingName
      Value: test-core
  AssociatedData:
  - Name: DataSourceTrace
    Description: Name of the data source
    Data:
    - OPC-UA Server
  - Name: EndpointUri
    Description: The endpoint to which the connection was attempted.
    Data:
    - '"opc.tcp://10.0.0.1:1234"'
```

Licenze

Questo componente è rilasciato in base al contratto di licenza del software Greengrass Core.

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
2.4.2	Correzioni di bug e miglioramenti • Risolve i problemi durante il rilevamento del server OPC UA in cui un nodo può essere scoperto più volte. • Corregge la funzionalità di istantanea per garantire che il timestamp sia nuovo per ogni punto dati dell'istantanea.
2.4.1	Correzioni di bug e miglioramenti • Risolve i problemi relativi al supporto proxy. • Risolve il problema in cui la pulizia del thread non riusciva e causava un blocco dei dati.
2.4.0	Nuove funzionalità • Aggiunge un registro degli eventi per facilitare l'identificazione e la risoluzione dei problemi. Correzioni di bug e miglioramenti • Risolve un problema con il client OPC-UA che causava errori di certifica to durante la connessione a un server OPC-UA che utilizza la versione 1.05 della specifica OPC-UA.
2.3.0	Nuove funzionalità • Aggiunge il supporto per la configurazione del proxy HTTP Greengrass nucleus su Linux. Correzioni di bug e miglioramenti • Risolve un problema di sicurezza (CVE-2019-19135).

2.2.0	Nuove funzionalità
	• Aggiunge il supporto per l'installazione di Data Collection Pack sull'arch itettura Linux ARMv8.
	• Requisiti minimi per Linux ARMv8:
	  • Memoria: 4 GB
	  • CPU: ARM Cortex-A72 o specifiche equivalenti

Versione	Modifiche
	Correzioni di bug e miglioramenti • Migliora la registrazione delle metriche nel processo di scoperta dei nodi. • Migliora la gestione dei tipi di dati non supportati. • Migliora la registrazione degli errori del flusso di dati.
2.1.3	Nuove funzionalità • Aggiunge il supporto per Windows Server 2019 o versioni successive. Correzioni di bug e miglioramenti • Migliora i messaggi di errore quando si distribuisce questo componente su dispositivi non supportati.

Versione	Modifiche

2.1.1	Nuove funzionalità
	• Aggiunge il supporto per la configurazione delle seguenti proprietà di sottoscrizione:
	  • DataChangeTrigger- È possibile definire la condizione che avvia un avviso di modifica dei dati.
	  • QueueSize- La profondità della coda su un server OPC-UA per una particolare metrica in cui sono messe in coda le notifiche per gli elementi monitorati.
	  • PublishingIntervalMilliseconds- L'intervallo (in millisecondi) di un ciclo di pubblicazione specificato al momento della creazione di un abbonamento.
	  • SnapshotFrequencyMilliseconds - È possibile configurare l'impostazione del timeout della frequenza delle istantanee per garantire che AWS IoT SiteWise Edge acquisisca un flusso costante di dati.
	• Questa versione supporta l'ingestione di dati di BAD qualità e filtra i dati in base alle seguenti qualità di dati:
	  • `UNCERTAIN` dati di qualità
	  • `BAD`dati di qualità
	Correzioni di bug e miglioramenti
	• Miglioramenti alle metriche dei clienti.
	• Risolve la codifica di sicurezza che a volte causava problemi durante la connessione a server con crittografia abilitata.
	• Risolve un problema a causa del quale il gruppo di proprietà non riusciva ad aggiornarsi.

2.0.3	Correzioni di bug e miglioramenti.
2.0.2	Correzioni di bug e miglioramenti alla sincronizzazione delle priorità delle risorse con Edge.
2.0.1	Versione iniziale.

Consulta anche

- Che cos'è? AWS IoT SiteWise nella Guida AWS IoT SiteWise per l'utente.

Simulatore di sorgenti dati IoT SiteWise OPC-UA

Il componente IoT SiteWise OPC-UA Data Source Simulator (`aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator`) avvia un server OPC-UA locale che genera dati di esempio. Usa questo server OPC-UA per simulare una fonte di dati letta dal componente del collettore IoT SiteWise OPC-UA su un gateway. AWS IoT SiteWise Quindi, puoi esplorare AWS IoT SiteWise le funzionalità utilizzando questi dati di esempio. Per ulteriori informazioni sui AWS IoT SiteWise gateway, consulta Using AWS IoT SiteWise at the edge nella Guida per l'AWS IoT SiteWiseutente.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 1.0.x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Il dispositivo principale Greengrass deve essere in grado di utilizzare la porta 4840 sull'host locale. Il server OPC-UA locale di questo componente funziona su questa porta.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle sue dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console. AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

La tabella seguente elenca le dipendenze per tutte le versioni di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2.3.0 <3.0.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente non ha parametri di configurazione.

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/
  aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs
  \aws.iot.SiteWiseEdgeOpcuaDataSourceSimulator.log -Tail 10 -Wait
  ```

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
1.0.0	Versione iniziale.

Versione	Modifiche
	Aggiunge il supporto per Windows Server 2016 o versioni successive.

Consulta anche

- Che cos'èAWS IoT SiteWise? nella Guida AWS IoT SiteWise per l'utente.

SiteWise Editore IoT

Il componente IoT SiteWise Publisher (`aws.iot.SiteWiseEdgePublisher`) consente ai AWS IoT SiteWise gateway di esportare dati dall'edge al Cloud AWS.

Per ulteriori informazioni sui AWS IoT SiteWise gateway, vedere Using AWS IoT SiteWise at the edge nella Guida per l'AWS IoT SiteWise utente.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 3.1.x

- 3.0.x
- 2.4.x
- 2.3.x
- 2.2.x
- 2.1.x
- 2,0x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Il dispositivo principale Greengrass deve funzionare su una delle seguenti piattaforme:
  - Sistema operativo: Ubuntu 18.04 o versione successiva

    Architettura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - Sistema operativo: Red Hat Enterprise Linux (RHEL) 8

    Architettura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - Sistema operativo: Amazon Linux 2

    Architettura: x86_64 (AMD64) o ARMv8 (Aarch64)
  - Sistema operativo: Debian 11

Architettura: x86_64 (AMD64) o ARMv8 (Aarch64)

- Sistema operativo: Windows Server 2019 o versione successiva

  Architettura: x86_64 (AMD64)

- Il dispositivo principale Greengrass deve connettersi a Internet.
- Il dispositivo principale Greengrass deve essere autorizzato a eseguire l'azione`iotsitewise:BatchPutAssetPropertyValue`. Per ulteriori informazioni, consulta Autorizzare i dispositivi principali a interagire con AWS i servizi.

  Example policy di autorizzazioni

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iotsitewise:BatchPutAssetPropertyValue",
            "Resource": "*"
        }
    ]
}
```

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`data.iotsitewise.` *`region`*`.amazonaws.com`	443	Sì	Pubblica dati su. AWS IoT SiteWise

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

La tabella seguente elenca le dipendenze per le versioni da 2.0.x a 2.2.x di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Nucleo Greengrass	>=2,3,0<3,0,0	Rigidi
Stream manager	>=2.3.0 =2,010<3,00	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente non ha parametri di configurazione.

Puoi utilizzare la AWS IoT SiteWise console o l'API per configurare il componente SiteWise editore IoT. Per ulteriori informazioni, consulta Fase 3: Configurazione dell'editore - opzionale nella Guida AWS IoT SiteWise per l'utente.

Dati di input

Questo componente legge i `PutAssetPropertyValueEntry` messaggi dallo AWS IoT Greengrass stream manager. Per ulteriori informazioni, consulta PutAssetPropertyValueEntryl'AWS IoT SiteWise API Reference.

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgePublisher.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgePublisher.log -Tail 10 -
  Wait
  ```

Risoluzione dei problemi e debug

Questo componente include un nuovo registro degli eventi per aiutare i clienti a identificare e risolvere i problemi. Il file di registro è separato dal file di registro locale e si trova nella seguente posizione. Sostituire */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

Linux

```
/greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/logs/
IotSiteWisePublisherEvents.log
```

Windows

```
C:\greengrass\v2\work\aws.iot.SiteWiseEdgePublisher\logs
\IotSiteWisePublisherEvents.log
```

Questo registro include informazioni dettagliate e istruzioni per la risoluzione dei problemi. Oltre alla diagnostica, vengono fornite informazioni sulla risoluzione dei problemi, con una descrizione di come risolvere il problema e, a volte, con collegamenti a ulteriori informazioni. Le informazioni diagnostiche includono quanto segue:

- Livello di gravità
- Timestamp
- Informazioni aggiuntive specifiche sull'evento

Example Log di esempio

```
accountBeingThrottled:
  Summary: Data upload speed slowed due to quota limits
  Level: WARN
  Timestamp: '2023-06-09T21:30:24.654Z'
  Description: The IoT SiteWise Publisher is limited to the "Rate of data points
 ingested"
    quota for a customers account. See the associated documentation and associated
    metric for the number of requests that were limited for more information. Note
    that this may be temporary and not require any change, although if the issue
 continues
    you may need to request an increase for the mentioned quota.
  FurtherInformation:
  - https://docs.aws.amazon.com/iot-sitewise/latest/userguide/quotas.html
  - https://docs.aws.amazon.com/iot-sitewise/latest/userguide/troubleshooting-
gateway.html#gateway-issue-data-streams
  AssociatedMetrics:
  - Name: TotalErrorCount
    Description: The total number of errors of this type that occurred.
    Value: 327724.0
  AssociatedData:
  - Name: AggregatePropertyAliases
    Description: The aggregated property aliases of the throttled data.
```

```
    FileLocation: /greengrass/v2/work/aws.iot.SiteWiseEdgePublisher/./logs/data/
AggregatePropertyAliases_1686346224654.log
```

Licenze

Questo componente è rilasciato in base al contratto di licenza del software Greengrass Core.

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
3.1.2	Correzioni di bug e miglioramenti • Risolto il problema dell'elevato utilizzo della CPU introdotto nella versione 3.1.1.
3.1.1	Correzioni di bug e miglioramenti • Aggiunge una registrazione aggiuntiva che identifica gli alias di dati interessati quando si verifica un errore. • Aggiunge l'applicazione locale dei limiti dell' AWS IoT SiteWise API sull'età dei dati acquisiti. • Risolve il problema per cui Publisher confonde i checkpoint degli StreamManager stream quando ci sono più destinazioni Amazon S3. • Risolve il problema delle prestazioni legato al modo in cui Publisher legge gli stream. StreamManager
3.1.0	Nuove funzionalità • Aggiunge il supporto per la pubblicazione di dati come file parquet su Amazon S3. • Aggiunge il supporto per l' AWS IoT SiteWise ingestione bufferizzata.
3.0.0	Correzioni di bug e miglioramenti • Risolve i problemi relativi al supporto proxy. Nuove funzionalità • Abilita il supporto per l'inserimento di dati da un broker MQTT.

Versione	Modifiche
2.4.1	Correzioni di bug e miglioramenti • Abilita il componente per funzionare con Java Corretto 11 versioni 11.0.20.8.1 e successive. Le versioni dei componenti 2.4.0 e 2.3.3 mostrano il messaggio di `"Could not find or load main class"` errore quando vengono utilizzate con la versione 11.0.20.8.1 di Java Corretto.
2.4.0	Nuove funzionalità • Aggiunge un nuovo registro degli eventi per facilitare l'identificazione e la risoluzione dei problemi. Correzioni di bug e miglioramenti • Migliora il ripristino del checkpoint di Publisher.
2.3.3	Correzioni di bug e miglioramenti • Migliora la capacità di supportare un throughput elevato.
2.3.2	Correzioni di bug e miglioramenti • Risolve il supporto del proxy HTTP durante il download della configura zione di Publisher.

2.3.1	Nuove funzionalità
	• Aggiunge il supporto per l'installazione di Data Collection Pack sull'arch itettura Linux ARMv8.
	• Requisiti minimi per Linux ARMv8:
	  • Memoria: 4 GB
	  • CPU: ARM Cortex-A72 o specifiche equivalenti

2.2.3	Correzioni di bug e miglioramenti • Rimuove i tentativi per un'eccezione generica che non era nell'elenco delle eccezioni recuperabili.

Versione	Modifiche
2.2.2	Correzioni di bug e miglioramenti • Reintroduce il supporto per il caricamento dei dati AWS IoT SiteWise tramite un server proxy HTTP.
2.2.1	**Note** Questa versione non supporta la configurazione del proxy HTTP. La versione 2.2.2 e successive reintroducono il supporto per questa funzionalità. Nuove funzionalità • Aggiunge il supporto a questo componente per attivare la compressione durante il caricamento dei dati su. AWS IoT SiteWise

Versione	Modifiche

2.2.0	**Note**
	Questa versione non supporta la configurazione del proxy HTTP. La versione 2.2.2 e successive reintroducono il supporto per questa funzionalità.
	
	Nuove funzionalità
	• Aggiorna questo componente per comprimere i dati prima di inviarli al servizio. AWS IoT SiteWise
	  • Nella maggior parte dei casi, questa modifica riduce l'utilizzo della larghezza di banda del 75% rispetto alle versioni precedenti di questo componente.
	  • Nella maggior parte dei casi, questa modifica aumenta l'utilizzo della CPU fino al 5%. Sui gateway che elaborano grandi quantità di dati, questa modifica può aumentare l'utilizzo della CPU fino al 15%.
	  • Questa modifica non influisce sui costi di AWS IoT SiteWise servizio o sull'utilizzo delle quote di servizio.
	  • Aggiunge il supporto per Windows Server 2019 o versioni successive.
	Correzioni di bug e miglioramenti
	• Risolve un problema che impedisce a questo componente di avviarsi quando il file di checkpoint è danneggiato.

2.1.4	Correzioni di bug e miglioramenti • Risolve la compatibilità con la versione Java 8.

Versione	Modifiche
2.1.3	**Warning** Questa versione non è più disponibile, tranne nelle regioni Stati Uniti orientali (Ohio), Canada (Centrale) e AWS GovCloud (Stati Uniti orientali). Questa versione del componente richiede la versione Java 11 o successiva per funzionare. I miglioramenti di questa versione sono disponibili nelle versioni successive di questo componente. Correzioni di bug e miglioramenti • Migliora i messaggi di errore quando si distribuisce questo componente su dispositivi non supportati. • Aggiorna per registrare gli errori quando il caricamento dei dati non riesce.
2.1.2	Correzioni di bug e miglioramenti • Aggiornamenti per richiamare la funzionalità di esportazione dei dati scaduti non appena i dati scadono.
2.1.1	Correzioni di bug e miglioramenti.
2.1.0	Nuove funzionalità • Aggiunge innanzitutto il supporto per la pubblicazione dei dati più recenti sul cloud. • Aggiunge il supporto per non pubblicare dati scaduti nel cloud. • Aggiunge il supporto per l'archiviazione locale dei dati scaduti. Correzioni di bug e miglioramenti • Riduce l'I/O del disco e la latenza corrispondente.
2.0.2	Correzioni di bug e miglioramenti.
2.0.1	Versione iniziale.

Consulta anche

- Che cos'è AWS IoT SiteWise? nella Guida AWS IoT SiteWise per l'utente.

SiteWise Processore IoT

Il componente SiteWise del processore IoT (`aws.iot.SiteWiseEdgeProcessor`) consente ai AWS IoT SiteWise gateway di elaborare i dati all'edge.

Con questo componente, i AWS IoT SiteWise gateway possono utilizzare modelli e asset di asset per elaborare i dati sui dispositivi gateway. Per ulteriori informazioni sui AWS IoT SiteWise gateway, consulta Using AWS IoT SiteWise at the edge nella Guida per l'AWS IoT SiteWise utente.

Argomenti

Versioni

Questo componente ha le seguenti versioni:

- 3.2.x
- 3.1.x
- 3.0.x
- 2.2.x
- 2.1.x
- 2,0x

Type

Questo componente è un componente generico () `aws.greengrass.generic`. Il nucleo Greengrass esegue gli script del ciclo di vita del componente.

Per ulteriori informazioni, consulta Tipi di componenti.

Sistema operativo

Questo componente può essere installato su dispositivi principali che eseguono i seguenti sistemi operativi:

- Linux
- Windows

Requisiti

Questo componente presenta i seguenti requisiti:

- Il dispositivo principale Greengrass deve funzionare su una delle seguenti piattaforme:
  - Sistema operativo: Ubuntu 20.04 o 18.04

    Architettura: x86_64 (AMD64)
  - Sistema operativo: Red Hat Enterprise Linux (RHEL) 8

    Architettura: x86_64 (AMD64)
  - Sistema operativo: Amazon Linux 2

    Architettura: x86_64 (AMD64)
  - Sistema operativo: Windows Server 2019 o versione successiva

    Architettura: x86_64 (AMD64)
- Il dispositivo principale Greengrass deve consentire il traffico in entrata sulla porta 443.
- Il dispositivo core Greengrass deve consentire il traffico in uscita sulle porte 443 e 8883.
- Le seguenti porte sono riservate all'uso da AWS IoT SiteWise: 80, 443, 3001, 4569, 4572, 8000, 8081, 8082, 8084, 8085, 8086, 8445, 9000, 9500, 11080 e 50010. L'utilizzo di una porta riservata per il traffico può comportare l'interruzione della connessione.

> **Note**
>
> La porta 8087 è richiesta solo per la versione 2.0.15 e successive di questo componente.

- Il ruolo del dispositivo Greengrass deve disporre di autorizzazioni che consentano di utilizzare i AWS IoT SiteWise gateway sui dispositivi. AWS IoT Greengrass V2 Per ulteriori informazioni, consulta Requisiti nella Guida per l'AWS IoT SiteWise utente.

Endpoint e porte

Questo componente deve essere in grado di eseguire richieste in uscita verso i seguenti endpoint e porte, oltre agli endpoint e alle porte necessari per le operazioni di base. Per ulteriori informazioni, consulta Consenti il traffico dei dispositivi tramite un proxy o un firewall.

Endpoint	Porta	Richiesto	Descrizione
`model.iotsitewise.region.amazonaws.com`	443	Sì	Ottieni informazioni sui tuoi AWS IoT SiteWise asset e sui tuoi modelli di asset.
`edge.iotsitewise.region.amazonaws.com`	443	Sì	Ottieni informazioni sulla configurazione del AWS IoT SiteWise gateway del

Endpoint	Porta	Richiesto	Descrizione
			dispositivo principale.
`ecr.region.amazonaws.com`	443	Sì	Scarica le immagini Docker del gateway AWS IoT SiteWise Edge da Amazon Elastic Container Registry.
`iot.region.amazonaws.com`	443	Sì	Ottieni gli endpoint dei dispositivi per il tuo Account AWS
`sts.region.amazonaws.com`	443	Sì	Ottieni l'ID del tuo Account AWS.

Endpoint	Porta	Richiesto	Descrizione
`monitor.iotsitewise. region.amazonaws.com`	443	No	Obbligatorio se accedi ai AWS IoT SiteWise Monitor portali sul dispositivo principale.

Dipendenze

Quando si distribuisce un componente, distribuisce AWS IoT Greengrass anche versioni compatibili delle relative dipendenze. Ciò significa che è necessario soddisfare i requisiti per il componente e tutte le sue dipendenze per distribuire correttamente il componente. Questa sezione elenca le dipendenze per le versioni rilasciate di questo componente e i vincoli di versione semantica che definiscono le versioni dei componenti per ogni dipendenza. È inoltre possibile visualizzare le dipendenze per ogni versione del componente nella console.AWS IoT Greengrass Nella pagina dei dettagli del componente, cerca l'elenco delle dipendenze.

La tabella seguente elenca le dipendenze per le versioni da 2.0.x a 2.1.x di questo componente.

Dipendenza	Versioni compatibili	Tipo di dipendenza
Servizio di scambio di token	>=2.0.3 <3.0.0	Rigidi
Stream manager	>=2,0,3 =2,0,10 <3,0,0	Rigidi
Greengrass CLI	>=2.3.0 <3.0.0	Rigidi

Per ulteriori informazioni sulle dipendenze dei componenti, vedere il riferimento alla ricetta dei componenti.

Configurazione

Questo componente non ha parametri di configurazione.

File di registro locale

Questo componente utilizza il seguente file di registro.

Linux

```
/greengrass/v2/logs/aws.iot.SiteWiseEdgeProcessor.log
```

Windows

```
C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeProcessor.log
```

Per visualizzare i log di questo componente

- Esegui il seguente comando sul dispositivo principale per visualizzare il file di registro di questo componente in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux

  ```
  sudo tail -f /greengrass/v2/logs/aws.iot.SiteWiseEdgeProcessor.log
  ```

  Windows (PowerShell)

  ```
  Get-Content C:\greengrass\v2\logs\aws.iot.SiteWiseEdgeProcessor.log -Tail 10 -
  Wait
  ```

Licenze

Questo componente include i seguenti software/licenze di terze parti:

- Apache 2.0
- NEBBIA
- Clausola BSD-2

- Clausola BSD-3
- CDDL-1.0
- CDDL-1.1
- DISCO
- Zlib
- GPL-3.0 con eccezione GCC
- Dominio pubblico
- Python 2.0
- Unicode DFS-2015
- Clausola BSD-1
- OpenSSL
- EPL-1.0
- EPL-2.0
- GPL-2.0- with-classpath-exception
- MPL-2,0
- CC0-1.0
- JSON

Questo componente è rilasciato in base al contratto di licenza del software Greengrass Core.

Changelog

La tabella seguente descrive le modifiche apportate a ciascuna versione del componente.

Versione	Modifiche
3.2.0	Miglioramenti in termini di prestazioni. • Ottimizza i servizi API per ridurre l'ingombro di memoria e richiedere meno spazio su disco per l'installazione • Ciò offre una riduzione di 2 GB nell'utilizzo iniziale della memoria (ora utilizza 7,5 GB di memoria all'avvio, tuttavia si consigliano comunque 16 GB) e una riduzione di 500 MB della dimensione del download (ora richiede un download di 1,4 GB) per l'intero componente.

Versione	Modifiche
	Nuove funzionalità • `GetAssetPropertyValueAggregates` L'API ora supporta finestre di aggregazione di 15 minuti sull'edge. • Le porte 8081 e 8082 non devono più essere disponibili per il corretto funzionamento di questo componente. **Note** L'endpoint locale per le API del piano AWS IoT SiteWise dati, ad esempio `get-asset-property-value` , viene modificato da a. `http://localhost:8081` `http://localhost:11080/data` L'endpoint locale per le API AWS IoT SiteWise del piano di controllo, ad esempio `list-asset-models` , viene modificato da a. `http://localhost:11080` `http://localhost:11080/control` AWS consiglia sempre di utilizzare gli endpoint HTTPS del gateway SiteWise Edge. Questi endpoint non sono cambiati. Correzioni di bug e miglioramenti • La sincronizzazione da ora AWS IoT SiteWise trasferirà le risorse in uno stato valido se la sincronizzazione precedente è stata interrotta. Ciò risolverà i problemi relativi al danneggiamento di alcune risorse dopo un riavvio forzato. • Risolve una rara condizione in cui una risorsa può essere danneggiata sul bordo se viene modificata durante la sincronizzazione. La sincroniz zazione ora fallirà se viene rilevata questa condizione e la risorsa verrà ritentata nella sincronizzazione successiva. • Risolve un problema che avrebbe potuto consentire la chiamata esterna dell'endpoint HTTP per le API. Ora è possibile utilizzare solo HTTPS per chiamare le API al di fuori dell'indirizzo di loopback locale. • `ListAssets` L'API ora mostra le gerarchie degli asset per gli asset archiviati sull'edge. • Risolve un problema per cui il Data Processing Pack non riusciva a riavviare, aggiornare o effettuare il downgrade su Windows.

Versione	Modifiche
	• Risolve un bug nel Data Processing Pack per il sistema operativo Windows che impediva ai clienti di utilizzare le credenziali per connetter si a un broker MQTT.
3.1.3	Correzioni di bug e miglioramenti • Risolve il problema per cui il Data Processing Pack segnalava erroneamente una sincronizzazione riuscita quando alcune risorse non funzionavano. • Consenti a più risorse di avere lo stesso nome purché non abbiano lo stesso elemento principale.
3.1.1	Correzioni di bug e miglioramenti • Risolve il problema in cui la richiesta SigV4 fallisce a causa di una mancata corrispondenza del fuso orario. • Risolto il problema per cui le proprietà di trasformazione e metrica interrompono il calcolo quando si basano sugli attributi dopo il riavvio. • Abilita il supporto della configurazione personalizzata delle porte di Stream Manager. • Risolve un problema per cui le proprietà sincronizzate con l'edge potrebbero smettere di essere aggiornate.
3.1.0	Correzioni di bug e miglioramenti • Risolve il problema in cui `ListAssetModels` l'API non riesce a generare il token successivo.
3.0.0	Nuove funzionalità • Abilita il supporto per l'inserimento di dati da un broker MQTT.

Versione	Modifiche
2.2.1	Correzioni di bug e miglioramenti • Modifica il processo di sincronizzazione per rendere l'archiviazione dei dati del piano di controllo più coerente con il funzionamento del cloud. Ciò influisce leggermente sull'aggiornamento. **Note** I dati del piano di controllo sincronizzati nella versione 2.2.1 o successiva non saranno compatibili con le versioni precedenti. Per effettuare il downgrade alle versioni precedenti, è necessari o completare una nuova installazione. Ciò non influisce sugli aggiornamenti, i dati sincronizzati nelle versioni precedenti funzioneranno con la versione 2.2.1. • Modifiche aggiuntive alla catena di credenziali per dare priorità alle AWS credenziali. AWS IoT Greengrass V2
2.1.37	Correzioni di bug e miglioramenti • Deprecate dependency-routing-service il processo e trasferitene le funzionalità nel property-state-service processo per ridurre l'utilizzo di risorse da parte dei processi che comunicano. • Aumenta il limite massimo di risultati per l'`get-asset-property-value-history` API a 20.000 in modo che corrisponda al limite utilizzato da. AWS IoT SiteWise • Risolve un problema per cui il token successivo non veniva fornito nei risultati impaginati per l'`get-asset-property-value-history` API quando non veniva specificato alcun limite massimo di risultati.
2.1.35	Correzioni di bug e miglioramenti • Modifica la catena di AWS credenziali per dare priorità alle credenziali. AWS IoT Greengrass • Risolve un problema relativo al rilevamento degli account durante la distribuzione come parte di un gruppo di oggetti. AWS IoT

Versione	Modifiche
2.1.34	Correzioni di bug e miglioramenti • Regola i calcoli metrici/trasformati per utilizzare il multithreading su Linux. Windows continua a eseguire calcoli a thread singolo per motivi di compatibilità. • Risolve un problema per cui i calcoli metrici mancavano in alcune finestre di calcolo.
2.1.33	Correzioni di bug e miglioramenti • Risolve un problema relativo alla segnalazione dello stato di errore nella console Greengrass.
2.1.32	Correzioni di bug e miglioramenti • Aggiunge il supporto per nomi utente e gruppi personalizzati.
2.1.31	Correzioni di bug e miglioramenti • Aggiunge il supporto per il calcolo della media ponderata nel tempo e della deviazione standard ponderata nel tempo per i dati modellati in. AWS IoT SiteWise
2.1.29	Correzioni di bug e miglioramenti • Aggiunge il supporto per filtrare le risorse sulla funzionalità edge.
2.1.28	Correzioni di bug e miglioramenti • Ottimizza la sincronizzazione delle risorse per consentire la sincroniz zazione di un gran numero di risorse dall'edge Cloud AWS all'edge.
2.1.24	Correzioni di bug e miglioramenti • Risolve un problema che causava la scomparsa della dashboard quando si sincronizzava una risorsa per la seconda volta.

Versione	Modifiche
2.1.23	Correzioni di bug e miglioramenti • È stato aggiunto un timeout per il processo di `aws.iot.SiteWiseEdgeProcessor` installazione per evitare errori di installazione se la connettività Internet è lenta. • Sincronizzazione ottimizzata delle risorse per migliorare l'efficienza di sincronizzazione tra cloud ed edge.
2.1.21	⚠ Warning L'aggiornamento da 2.0.x a 2.1.x comporterà la perdita di dati locali. Nuove funzionalità • Aggiunge il supporto per Windows Server 2019 o versioni successive. • Rimuove i docker per i sistemi operativi basati su Linux.
2.0.16	Questa versione contiene correzioni di bug e miglioramenti.
2.0.15	Correzioni di bug e miglioramenti • Modifica la porta utilizzata da questo componente per le operazioni dell'API di sincronizzazione delle risorse da 8085 a 8087. Di conseguenza, questo componente ora richiede che la porta 8087 sia disponibile. Questo componente richiede ancora che la porta 8085 sia disponibile. • Aggiorna AWS OpsHub l'autenticazione per negare l'accesso agli utenti non autorizzati anziché quando un utente tenta di richiamare le operazioni API.
2.0.14	Questa versione contiene correzioni di bug e miglioramenti.
2.0.13	Correzioni di bug e miglioramenti • Risolve un problema per cui quando questo componente riporta i dati alle CloudWatch metriche di Amazon, ora indica correttamente quali dati non sono modellati.

Versione	Modifiche
2.0.9	Correzioni di bug e miglioramenti • Migliora l'affidabilità per creare e aggiornare AWS IoT SiteWise risorse sul dispositivo principale. • Aggiunge operazioni API locali aggiuntive che è possibile utilizzare per monitorare quali componenti sono installati sul dispositivo principale, la versione di ciascun componente e lo stato di ciascun componente. È possibile visualizzare queste informazioni nella scheda Impostazioni dell' AWS IoT SiteWise applicazione AWS OpsHub for sul dispositivo principale. • Aggiunge uno stato di integrità per i contenitori Docker eseguiti da questo componente. Puoi eseguire il `docker ps` comando per visualizz are lo stato di integrità dei contenitori.
2.0.7	Correzioni di bug e miglioramenti • Risolve il supporto per la visualizzazione AWS IoT SiteWise Monitor dei portali sul dispositivo principale.
2.0.6	Correzioni di bug e miglioramenti • Corregge AWS IoT SiteWise `statetime()` le `latest()` funzioni e `earliest()` le funzioni che questo componente calcola sul dispositi vo principale.
2.0.5	Correzioni di bug e miglioramenti • Aggiunge il supporto per la AWS IoT SiteWise `pretrigger()` funzione nelle trasformazioni che questo componente calcola sul dispositivo principale. • Modifica il percorso in cui questo componente memorizza la configura zione LDAP (Lightweight Directory Access Protocol) per l'autenticazione.
2.0.2	Versione iniziale.

Consulta anche

- Che cos'è AWS IoT SiteWise? nella Guida AWS IoT SiteWise per l'utente.

Componenti supportati da Publisher

I componenti supportati da Publisher sono disponibili in una versione di anteprima e sono soggetti a modifiche. AWS IoT Greengrass Questi componenti non sono supportati da. AWS È necessario contattare l'editore per eventuali problemi relativi a ciascuno dei componenti.

I componenti supportati da Greengrass Publisher sono sviluppati, offerti e gestiti da fornitori di componenti di terze parti. I fornitori di componenti di terze parti provengono da AWS Partner Device Catalog, Heroes o dai fornitori della community. AWS Puoi acquistare i componenti di questo catalogo contattando direttamente il fornitore di componenti di terze parti.

I componenti supportati da Greengrass Publisher includono quanto segue:

Argomenti

- AIShield.edge
- EdgeLabs Sensore AI
- Greengrass S3 Ingestor

AIShield.edge

Questo componente è stato sviluppato ed è supportato da AiShield, con tecnologia Bosch. Migliora la tua sicurezza AI con AIShield.edge. Questo componente è progettato per implementare senza problemi difese personalizzate basate sulle minacce sui dispositivi periferici, che proteggono i dispositivi dagli attacchi di intelligenza artificiale.

Questo componente offre i seguenti vantaggi:

- Passa senza problemi dall'analisi delle vulnerabilità con AIShield AI Security alle difese edge fortificate interne AWS
- Implementa difese personalizzate su più dispositivi perimetrali con facilità
- Ampia protezione personalizzata per diverse configurazioni di intelligenza artificiale che supporta vari tipi di modelli e framework
- Resta aggiornato grazie alla perfetta integrazione nei flussi di Amazon SageMaker lavoro Greengrass

- Ottieni informazioni immediate sulle potenziali minacce, con i dati trasmessi direttamente a AWS IoT Core
- Un percorso di sicurezza AI coeso per l'implementazione della difesa sull'edge di AIShield AI Security on the Marketplace AWS

Questo componente deve essere eseguito sulla seguente piattaforma:

- Sistema operativo: Linux

Se siete interessati all'acquisto di questo componente, contattate Bosch Software and Digital Solutions: <AIShield.Contact@bosch.com>.

EdgeLabs Sensore AI

Questo componente è stato sviluppato ed è supportato dall'IA EdgeLabs. AI EdgeLabs Sensor è un'applicazione basata su container che contiene funzionalità di rilevamento e prevenzione delle minacce basate sull'intelligenza artificiale. AI Sensor è racchiuso in un componente Greengrass e distribuito come contenitore autonomo sul dispositivo principale insieme ad altri componenti Greengrass.

Questo componente attuale è un agente basato su container che verifica continuamente le comunicazioni di rete, cerca i modelli di minaccia nel software in esecuzione sull'Edge Host o sul gateway IoT. Questo componente utilizza eBPF, la verifica comportamentale della larghezza di banda dei processi e la configurazione basata su host. La funzionalità principale di questo componente si basa sulle funzioni NDR/IPS ed EDR.

Questo componente offre i seguenti vantaggi:

- Rilevamento delle minacce basato sull'intelligenza artificiale contro attacchi di rete e malware (EDR/NDR)
- Risposta agli incidenti automatizzata basata sull'intelligenza artificiale (IPS)
- Intelligence sulle minacce a livello locale dell'host con trasferimento minimo di dati all'esterno
- Implementazione leggera con Docker e Greengrass

Questo componente deve essere eseguito su una delle seguenti piattaforme:

- Sistema operativo: Linux

Se sei interessato all'acquisto di questo componente, contatta AI EdgeLabs: <contact@edgelabs.ai>.

Greengrass S3 Ingestor

Questo componente è stato sviluppato ed è supportato da Nathan Glover. Il componente Greengrass S3 Ingestor è progettato per essere utilizzato con il componente stream manager. Questo componente prende un flusso delimitato da righe di messaggi JSON dallo stream manager e li raggruppa in un file GZIP. Questo componente consente l'inserimento efficiente dei dati in Amazon S3 per ulteriori elaborazioni o per lo storage. Questo componente non supporta l'invio di dati in tempo reale. Cloud AWS

Questo componente deve essere eseguito su una delle seguenti piattaforme:

- Sistema operativo: Linux
- Sistema operativo: Windows

<Se sei interessato all'acquisto di questo componente, contatta Nathan Glover: n

Componenti comunitari

Il Greengrass Software Catalog è un indice dei componenti Greengrass sviluppati dalla comunità Greengrass. Da questo catalogo, puoi scaricare, modificare e distribuire componenti per creare le tue applicazioni Greengrass. È possibile visualizzare il catalogo al seguente link: https://github.com/aws-greengrass/. aws-greengrass-software-catalog

Ogni componente dispone di un GitHub archivio pubblico che puoi esplorare. Visualizza il catalogo del software Greengrass GitHub per trovare l'elenco completo dei componenti della community. Ad esempio, questo catalogo include i seguenti componenti:

- Flusso di video Amazon Kinesis

  Questo componente acquisisce flussi audio e video da telecamere locali che utilizzano il Real Time Streaming Protocol (RTSP). Il componente carica quindi i flussi audio e video su Amazon Kinesis Video Streams.
- Gateway Bluetooth IoT

  Questo componente utilizza la BluePylibreria che consente la comunicazione con i dispositivi Bluetooth Low Energy (LE) per creare interfacce client Bluetooth LE.

- Certificate Rotator

  Questo componente fornisce un mezzo per ruotare il certificato e la AWS IoT Greengrass chiave privata del dispositivo principale, in tutta la flotta, su larga scala.

- Tunneling sicuro containerizzato

  Questo componente fornisce un contenitore Docker per il tunneling sicuro con tutte le dipendenze e le librerie corrispondenti in una ricetta riutilizzabile che non si basa su uno specifico sistema operativo host.

- Grafana

  Questo componente consente di ospitare un server Grafana su un dispositivo core Greengrass. Puoi utilizzare le dashboard Grafana per visualizzare e gestire i dati sul dispositivo principale.

- GStreamer per Amazon Lookout for Vision

  Questo componente fornisce un plug-in GStreamer in modo da poter eseguire il rilevamento delle anomalie di Lookout for Vision nelle pipeline GStreamer personalizzate.

- Assistente domestico

  Questo componente consente al cliente di utilizzare Home Assistant per fornire il controllo locale dei dispositivi domestici intelligenti. Fornisce l'integrazione con AWS i servizi perimetrali e nel cloud per fornire soluzioni di automazione domestica che estendono Home Assistant.

- Dashboard InfluxDB/Grafana

  Questo componente offre un'esperienza con un clic per configurare i componenti InfluxDB e Grafana. Collega InfluxDB a Grafana e automatizza la configurazione di una dashboard Grafana locale che esegue il rendering della telemetria in tempo reale. AWS IoT Greengrass

- InfluxDB

  Questo componente fornisce un database di serie temporali InfluxDB su un dispositivo core Greengrass. È possibile utilizzare questo componente per elaborare i dati dai sensori IoT, analizzare i dati in tempo reale e monitorare le operazioni all'edge.

- Editore InfluxDB

  Questo componente trasmette la telemetria AWS IoT Greengrass sullo stato del sistema dal plug-in Nucleus emitter a InfluxDB. Questo componente può anche inoltrare telemetria personalizzata a InfluxDB.

- Framework pubsub IoT

Questo framework fornisce un'architettura applicativa, un codice modello ed esempi implementabili che aiutano a migliorare la qualità del codice per le applicazioni pubsub IoT distribuite basate sugli eventi che utilizzano componenti personalizzati v2. AWS IoT Greengrass Per ulteriori informazioni, consulta Crea AWS IoT Greengrass componenti.

- Jupyter Labs

  Questo componente viene distribuito su un dispositivo principale. JupyterLab AWS IoT Greengrass L'ambiente Jupyter ha accesso alle risorse variabili di processo e ambiente impostate da AWS IoT Greengrass, semplificando il processo di test e sviluppo di componenti scritti in Python.

- Server web locale

  Questo componente consente di creare un'interfaccia utente Web locale su un dispositivo principale Greengrass. È possibile creare un'interfaccia utente Web locale che consenta, ad esempio, di configurare le impostazioni del dispositivo e dell'applicazione o di monitorare il dispositivo.

- LoRaWaAdattatore di protocollo N

  Questo componente acquisisce dati da dispositivi wireless locali che utilizzano il protocollo LoRaWa N, che è un protocollo LPWAN (Low Power Wide Area Network). Il componente consente di analizzare e agire sui dati localmente senza comunicare con il cloud.

- Modbus TCP

  Questo componente raccoglie dati dai dispositivi locali utilizzando il protocollo ModbusTCP e li pubblica su flussi di dati selezionati.

- Node-red

  Questo componente installa Node-RED su un AWS IoT Greengrass dispositivo principale utilizzando NPM. Il componente dipende dal componente Node-RED Auth che deve essere distribuito e configurato in modo esplicito. Puoi utilizzare la CLI Node-RED per Greengrass per distribuire i flussi Node-RED sui dispositivi. AWS IoT Greengrass

- Node-RED Docker

  Questo componente installa Node-RED sul dispositivo AWS IoT Greengrass principale utilizzando il contenitore Docker Node-RED ufficiale. Il componente dipende dal componente Node-RED Auth che deve essere distribuito e configurato in modo esplicito. Puoi utilizzare la CLI Node-RED per Greengrass per distribuire i flussi Node-RED sui dispositivi. AWS IoT Greengrass

- Autenticazione Node-RED

  Questo componente configura un nome utente e una password per proteggere l'istanza Node-RED in esecuzione su un dispositivo principale. AWS IoT Greengrass

- OpenThreadRouter di frontiera

  Questo componente distribuisce il contenitore OpenThread Border Router Docker. Il componente aiuta a comporre un dispositivo Matter che include un router di confine Thread.

- Connettore dati di streaming OSI Pi

  Questo componente fornisce lo streaming di dati in tempo reale da OSI Pi Data Archive a una moderna architettura di dati. AWS Si integra con OSI Pi Asset Framework, gestito centralmente tramite messaggistica. AWS IoT PubSub

- Provider Parsec

  Questo componente consente ai AWS IoT Greengrass dispositivi di integrare soluzioni di sicurezza hardware utilizzando il progetto Parsec open source di Cloud Native Computing Foundation (CNCF).

- Database PostgreSQL

  Questo componente fornisce supporto per il database relazionale PostgreSQL all'edge. I clienti possono utilizzare questo componente per fornire e gestire un'istanza PostgreSQL locale all'interno di un contenitore docker.

- Caricatore di file S3

  Questo componente monitora una directory alla ricerca di nuovi file, li carica su Amazon Simple Storage Service (Amazon S3) e quindi li elimina dopo un caricamento riuscito.

- Cliente Secrets Manager

  Questo componente fornisce uno strumento CLI che può essere utilizzato da altri componenti che devono recuperare segreti dal componente Secrets Manager in uno script del ciclo di vita della ricetta.

- Instradamento TES verso il contenitore

  Questo componente configura nftables o iptables su un AWS IoT Greengrass dispositivo in modo che possa utilizzare il componente con i contenitori. Servizio di scambio di token

- WebRTC

Questo componente acquisisce flussi audio e video dalle telecamere RTSP collegate al dispositivo principale. AWS IoT Greengrass Quindi il componente trasforma i flussi audio e video in peer-to-peer comunicazione o inoltro tramite Amazon Kinesis Video Streams.

Per richiedere una funzionalità o segnalare un bug, apri un GitHub problema nell'archivio di quel componente. AWS non fornisce supporto per i componenti della community. Per ulteriori informazioni, consulta il CONTRIBUTING.mdfile nel repository di ogni componente.

Diversi componenti AWS forniti sono inoltre open source. Per ulteriori informazioni, consulta Software AWS IoT Greengrass Core open source.

AWS IoT Greengrassstrumenti di sviluppo

Usa gli strumenti di AWS IoT Greengrass sviluppo per creare, testare, creare, pubblicare e distribuire componenti Greengrass personalizzati.

- CLI del kit di sviluppo Greengrass

  Utilizzate l'interfaccia a riga di comando del AWS IoT Greengrass Development Kit (GDK CLI) nel vostro ambiente di sviluppo locale per creare componenti da modelli e componenti della community nel Greengrass Software Catalog. Puoi usare la CLI GDK per creare il componente e pubblicarlo nel servizio come componente privato AWS IoT Greengrass del tuo. Account AWS
- Interfaccia a riga di comando Greengrass

  Utilizza l'interfaccia a riga di comando Greengrass (Greengrass CLI) sui dispositivi principali Greengrass per distribuire ed eseguire il debug dei componenti Greengrass. La CLI di Greengrass è un componente che puoi distribuire sui tuoi dispositivi principali per creare distribuzioni locali, visualizzare dettagli sui componenti installati ed esplorare i file di registro.
- Console di debug locale

  Utilizza la console di debug locale sui dispositivi principali Greengrass per distribuire ed eseguire il debug dei componenti Greengrass utilizzando un'interfaccia web del dashboard locale. La console di debug locale è un componente che puoi distribuire sui tuoi dispositivi principali per creare distribuzioni locali e visualizzare i dettagli sui componenti installati.

AWS IoT Greengrassfornisce anche i seguenti SDK che è possibile utilizzare nei componenti Greengrass personalizzati:

- IISDK per dispositivi AWS IoT, che contiene la libreria di comunicazione tra processi (IPC). Per ulteriori informazioni, consulta Usa il SDK per dispositivi AWS IoT per comunicare con il nucleo Greengrass, altri componenti e AWS IoT Core.
- L'SDK Stream Manager, che è possibile utilizzare per trasferire flussi di dati verso. Cloud AWS Per ulteriori informazioni, consulta Gestisci i flussi di dati sui dispositivi core Greengrass.

Argomenti

- AWS IoT GreengrassInterfaccia a riga di comando del Development Kit
- Interfaccia a riga di comando Greengrass
- Usa AWS IoT Greengrass Testing Framework

AWS IoT GreengrassInterfaccia a riga di comando del Development Kit

L'interfaccia a riga di comando del AWS IoT Greengrass Development Kit (GDK CLI) fornisce funzionalità che aiutano a sviluppare componenti Greengrass personalizzati. Puoi usare la CLI GDK per creare, creare e pubblicare componenti personalizzati. Quando crei un repository di componenti con la CLI GDK, puoi iniziare da un modello o da un componente della community dal Greengrass Software Catalog. Quindi, puoi scegliere un sistema di compilazione che impacchetti i file come archivi ZIP, utilizzi uno script di compilazione Maven o Gradle o esegua un comando di compilazione personalizzato. Dopo aver creato un componente, puoi utilizzare la CLI di GDK per pubblicarlo sul servizio, in modo da poter utilizzare AWS IoT Greengrass la console o AWS IoT Greengrass l'API per distribuire il componente sui tuoi dispositivi principali Greengrass.

Quando sviluppate componenti Greengrass senza la CLI GDK, dovete aggiornare la versione e gli URI degli artefatti nel file di ricetta del componente ogni volta che create una nuova versione del componente. Quando usi la CLI di GDK, può aggiornare automaticamente la versione e gli URI degli artefatti ogni volta che pubblichi una nuova versione del componente.

La CLI GDK è open source e disponibile su. GitHub Puoi personalizzare ed estendere la CLI GDK per soddisfare le tue esigenze di sviluppo dei componenti. Ti invitiamo ad aprire problemi e scaricare richieste dal repository. GitHub Puoi trovare il codice sorgente della CLI di GDK al seguente link: https://github.com/aws-greengrass/. aws-greengrass-gdk-cli

Prerequisiti

Per installare e utilizzare la CLI del Greengrass Development Kit, è necessario quanto segue:

- Un Account AWS. Se non lo hai, consultare Configura un Account AWS.
- Un computer di sviluppo simile a Windows, macOS o UNIX con una connessione Internet.
- Per la versione 1.1.0 o successiva della CLI di GDK, Python 3.6 o successiva installato sul computer di sviluppo.

  Per la versione 1.0.0 della CLI di GDK, Python 3.8 o successiva installata sul computer di sviluppo.
- Git installato sul tuo computer di sviluppo.
- AWS Command Line Interface(AWS CLI) installato e configurato con credenziali sul computer di sviluppo. Per ulteriori informazioni, vedere Installazione, aggiornamento e disinstallazione AWS CLI e configurazione di AWS CLI nella Guida per l'utente. AWS Command Line Interface

  > **Note**
  >
  > Se utilizzi un Raspberry Pi o un altro dispositivo ARM a 32 bit, installa V1. AWS CLI AWS CLI La versione 2 non è disponibile per i dispositivi ARM a 32 bit. Per ulteriori informazioni, vedere Installazione, aggiornamento e disinstallazione della AWS CLI versione 1.
- Per utilizzare la CLI GDK per pubblicare componenti sul servizio, AWS IoT Greengrass è necessario disporre delle seguenti autorizzazioni:
  - `s3:CreateBucket`
  - `s3:GetBucketLocation`
  - `s3:PutObject`
  - `greengrass:CreateComponentVersion`
  - `greengrass:ListComponentVersions`
- Per utilizzare la CLI GDK per creare un componente i cui artefatti esistono in un bucket S3 e non nel file system locale, è necessario disporre delle seguenti autorizzazioni:
  - `s3:ListBucket`

  Questa funzionalità è disponibile per GDK CLI v1.1.0 e versioni successive.

Changelog

La tabella seguente descrive le modifiche in ogni versione della CLI GDK. Per ulteriori informazioni, consulta la pagina dei rilasci della CLI di GDK su. GitHub

Versione	Modifiche
1.6.2	Correzioni di bug e miglioramenti • Risolve un problema per cui Windows gradlew.bat non funziona a causa del percorso relativo. • Piccoli miglioramenti alla registrazione, al test e alla creazione di pacchetti.
1.6.1	Correzioni di bug e miglioramenti • Aggiunge una correzione di sicurezza per l'analisi degli argomenti della CLI. • Consente a GDK di ottenere il nome della release più recente di Greengrass Testing Framework (GTF) come versione GTF predefinita. • Consente a GDK di consigliare ai clienti che utilizzano una versione precedente di GTF di aggiornare alla versione più recente.
1.6.0	Nuove funzionalità • Aggiunge un controllo di convalida della ricetta rispetto allo schema della ricetta Greengrass durante `component build` i `component publish` comandi and. Questo aggiornamento aiuta gli sviluppatori a identificare i problemi risolvibili all'interno delle ricette dei componenti nelle prime fasi del processo di creazione dei componenti. • Aggiunge una suite di test di confidenza al modello che può essere richiamata dal comando. `test-e2e init` Questa suite di test di fiducia include otto test generici che possono essere utilizzati ed estesi per soddisfare le esigenze di test dei componenti di base. Correzioni di bug e miglioramenti • Aggiorna la versione predefinita di Greengrass Testing Framework (GTF) utilizzata dal `test-e2e` comando alla versione 1.2.0.
1.5.0	Correzioni di bug e miglioramenti Aggiorna i modelli riconosciuti dall'opzione `excludes` build quando lo `build_system èzip`. Questa versione ora riconoscerà i pattern globulari che corrispondono ai nomi dei percorsi in base ai loro caratteri

Versione	Modifiche
	jolly. Ciò consente di specificare in modo personalizzato le directory da cui escludere.
1.4.0	Nuove funzionalità • Aggiunge un nuovo `config` comando che avvia un prompt interatti vo per modificare i campi all'interno di un file di configurazione GDK esistente. • Modifica i `gdk component publish` comandi `gdk component build` and per verificare che la dimensione della ricetta rientri nei requisiti di Greengrass (<=16000 byte) prima di procedere. Correzioni di bug e miglioramenti • Aggiunge una registrazione aggiuntiva nell'output del `gdk component build` comando quando un errore di sintassi della ricetta impedisce il completamento della compilazione per rendersene conto. • Rinomina `gtf-version` rispettivamente `otf-options` e `otf-version` in `gtf-options` e, a causa della ridenominazione di Open Test Framework in Greengrass Testing Framework.
1.3.0	Nuove funzionalità • Aggiunge un nuovo `test-e2e` comando per supportare il end-to-end test dei componenti utilizzando Open Test Framework. • Aggiunge una nuova opzione di configurazione per supportare nomi di file zip configurabili con il sistema di compilazione zip. `zip_name` • Rende `region` facoltativa la proprietà nel file di configurazione GDK. Correzioni di bug e miglioramenti • Risolve un problema per cui viene creata una nuova directory anche quando il modello o il repository specificato non esiste durante l'inizial izzazione di un progetto GDK con l'argomento. `--name`

Versione	Modifiche
1.2.3	Correzioni di bug e miglioramenti • Risolve un problema in cui la creazione del bucket non riesce a causa di una gestione errata degli errori. • Risolve un problema per cui le strutture degli elenchi nella ricetta del componente vengono rimosse.
1.2.2	Correzioni di bug e miglioramenti • I codici delle ricette non distinguono più tra maiuscole e minuscole. • Aggiunge un controllo per determinare se un bucket esiste in un bucket Regione AWS ed è accessibile dall'utente prima di creare un nuovo bucket. Richiede che l'utente disponga dell'`GetBucketLocation` autorizzazione. • Risolve un problema con la `excludes` parola chiave nel file di configurazione della CLI GDK.
1.2.1	Correzioni di bug e miglioramenti • Accetta la voce di configurazione della regione Canada (Central`ca-central-1`) () Regione AWS nel `gdk-config.json` file. • Risolve i problemi con l'argomento `--region` GDK CLI del comando. `publish`

Versione	Modifiche
1.2.0	Nuove funzionalità • Aggiunge la `options` voce alla `build` configurazione nel file di configurazione GDK CLI. Supporta `excludes` under `options` per escludere determinati file dall'artefatto zip quando si utilizza il sistema di compilazione. `zip` • Aggiunge il sistema di `gradlew` compilazione per utilizzare Gradle Wrapper per creare componenti. • Aggiunge il supporto per i file di build Kotlin DSL per l'opzione build. `gradle` • Aggiunge una `options` voce alla `publish` configurazione nel file di configurazione GDK CLI. Supporta quanto `file_upload_args` segue `options` per fornire argomenti aggiuntivi durante il caricamento di file su Amazon S3. Correzioni di bug e miglioramenti • Risolve un problema per cui le build di Gradle non venivano pulite prima di eseguire un comando di compilazione. • Risolve un problema per cui la build non usciva quando il comando build fallisce. • Migliora il formato di output del `gdk component list` comando.

Versione	Modifiche
1.1.0	Nuove funzionalità • Aggiunge il supporto per il sistema di build Gradle. • Aggiunge il supporto per il sistema di build Maven sui dispositivi Windows. • Aggiunge l'`--bucket`argomento al comando di pubblicazione del componente. Puoi usare questo argomento per specificare il bucket esatto in cui la CLI di GDK carica gli artefatti dei componenti. • Aggiunge l'`--name`argomento al comando component init. È possibile utilizzare questa opzione per specificare la cartella in cui la CLI GDK inizializza il componente. • Aggiunge il supporto per gli artefatti dei componenti che esistono in un bucket S3 ma non nella cartella di build del componente locale. Puoi utilizzare questa funzionalità per ridurre i costi della larghezza di banda per componenti di grandi dimensioni, come i modelli di machine learning. Correzioni di bug e miglioramenti • Aggiorna il comando di pubblicazione del componente per verificare se il componente è stato creato prima di pubblicare il componente. Se il componente non è stato creato, questo comando ora lo crea automaticamente. • Risolve un problema per cui il sistema di compilazione zip non riesce a creare su dispositivi Windows quando il nome del file ZIP contiene lettere maiuscole. • Migliora il formato dei messaggi di registro e modifica il livello di registro predefinito `INFO` su dispositivi che eseguono versioni di Python precedenti alla 3.8. • Modifica il requisito minimo della versione di Python in Python 3.6.
1.0.0	Versione iniziale.

Installare o aggiornare l'interfaccia a riga di comando del AWS IoT Greengrass Development Kit

L'interfaccia a riga di comando del AWS IoT Greengrass Development Kit (GDK CLI) è costruita su Python, quindi puoi `pip` usarla per installarla sul tuo computer di sviluppo.

> **Tip**
>
> Puoi anche installare la CLI GDK in ambienti virtuali Python come venv. Per ulteriori informazioni, consulta Ambienti e pacchetti virtuali nella documentazione di Python 3.

Per installare o aggiornare la CLI GDK

1. Esegui il comando seguente per installare l'ultima versione della CLI GDK dal GitHub suo repository.

   ```
   python3 -m pip install -U git+https://github.com/aws-greengrass/aws-greengrass-gdk-cli.git@v1.6.2
   ```

   > **Note**
   >
   > Per installare una versione specifica della CLI GDK, *`sostituisci`* versionTag con il tag di versione da installare. Puoi visualizzare i tag di versione per la CLI di GDK nel GitHub suo repository.
   >
   > ```
   > python3 -m pip install -U git+https://github.com/aws-greengrass/aws-greengrass-gdk-cli.git@versionTag
   > ```

2. Esegui il comando seguente per verificare che la CLI GDK sia stata installata correttamente.

   ```
   gdk --help
   ```

   Se il `gdk` comando non viene trovato, aggiungi la sua cartella a PATH.

   - Sui dispositivi Linux, aggiungilo `/home/`*`MyUser`*`/.local/bin` a PATH e sostituiscilo *`MyUser`*con il nome dell'utente.

- Sui dispositivi Windows, aggiungi *PythonPath*`\\Scripts` a PATH e sostituisci *PythonPath*con il percorso della cartella Python sul tuo dispositivo.

Ora puoi usare la CLI GDK per creare, creare e pubblicare componenti Greengrass. Per ulteriori informazioni su come utilizzare la CLI di GDK, vedere. AWS IoT GreengrassComandi dell'interfaccia a riga di comando del Development Kit

AWS IoT GreengrassComandi dell'interfaccia a riga di comando del Development Kit

L'interfaccia a riga di comando del AWS IoT Greengrass Development Kit (GDK CLI) fornisce un'interfaccia a riga di comando che puoi usare per creare, creare e pubblicare componenti Greengrass sul tuo computer di sviluppo. I comandi GDK CLI utilizzano il seguente formato.

```
gdk <command> <subcommand> [arguments]
```

Quando installi la CLI GDK, il programma di installazione si `gdk` aggiunge al PATH in modo da poter eseguire la CLI GDK dalla riga di comando.

È possibile utilizzare i seguenti argomenti con qualsiasi comando:

- Usa `-h` o `--help` per informazioni su un comando GDK CLI.
- Usa `-v` o `--version` per vedere quale versione di GDK CLI è installata.
- Usa `-d` o `--debug` per generare log dettagliati che puoi usare per eseguire il debug della CLI di GDK.

Questa sezione descrive i comandi GDK CLI e fornisce esempi per ogni comando. La sinossi di ogni comando mostra i relativi argomenti e il loro utilizzo. Gli argomenti opzionali sono indicati tra parentesi quadre.

Comandi disponibili

- componente
- config
- test-e2e

componente

Usa il `component` comando nell'interfaccia a riga di comando del AWS IoT Greengrass Development Kit (GDK CLI) per creare, creare e pubblicare componenti Greengrass personalizzati.

Sottocomandi

- init
- build
- pubblicazione
- elenco

init

Inizializza una cartella di componenti Greengrass da un modello di componente o da un componente della community.

La CLI GDK recupera i componenti della community dal Greengrass Software Catalog e i modelli di componenti dal AWS IoT Greengrass repository Component Templates in poi. GitHub

> **Note**
>
> Se usi GDK CLI v1.0.0, devi eseguire questo comando in una cartella vuota. La CLI GDK scarica il modello o il componente della community nella cartella corrente.
> Se utilizzi GDK CLI v1.1.0 o versione successiva, puoi specificare `--name` l'argomento per specificare la cartella in cui la CLI di GDK scarica il modello o il componente della community. Se usi questo argomento, specifica una cartella che non esiste. La CLI GDK crea la cartella per te. Se non specificate questo argomento, la CLI di GDK utilizza la cartella corrente, che deve essere vuota.
> Se il componente utilizza il sistema di compilazione zip, la CLI di GDK comprime determinati file nella cartella del componente in un file zip con lo stesso nome della cartella del componente. Ad esempio, se il nome della cartella del componente è`HelloWorld`, la CLI GDK crea un file zip denominato. `HelloWorld.zip` Nella ricetta del componente, il nome dell'artefatto zip deve corrispondere al nome della cartella del componente. Se si utilizza la versione 1.0.0 della CLI di GDK su un dispositivo Windows, i nomi delle cartelle dei componenti e dei file zip devono contenere solo lettere minuscole.
> Se inizializzi un modello o un componente della community che utilizza il sistema di compilazione zip in una cartella con un nome diverso dal modello o dal componente, devi modificare il nome dell'artefatto zip nella ricetta del componente. Aggiorna le `Lifecycle`

definizioni `Artifacts` and in modo che il nome del file zip corrisponda al nome della cartella del componente. L'esempio seguente evidenzia il nome del file zip nelle `Lifecycle` definizioni `Artifacts` and.

JSON

```
{
  ...
  "Manifests": [
    {
      "Platform": {
        "os": "all"
      },
      "Artifacts": [
        {
          "URI": "s3://BUCKET_NAME/COMPONENT_NAME/
COMPONENT_VERSION/HelloWorld.zip",
          "Unarchive": "ZIP"
        }
      ],
      "Lifecycle": {
        "run": "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
 {configuration:/Message}"
      }
    }
  ]
}
```

YAML

```
---
...
Manifests:
  - Platform:
      os: all
    Artifacts:
      - URI: "s3://BUCKET_NAME/COMPONENT_NAME/
COMPONENT_VERSION/HelloWorld.zip"
        Unarchive: ZIP
    Lifecycle:
```

```
        run: "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
    {configuration:/Message}"
```

Riepilogo

```
$ gdk component init
    [--language]
    [--template]
    [--repository]
    [--name]
```

Argomenti (inizializzati dal modello del componente)

- `-l`, `--language` — Il linguaggio di programmazione da utilizzare per il modello specificato.

  È necessario specificare `--repository` o `--language` e`--template`.

- `-t`, `--template` — Il modello di componente da utilizzare per un progetto di componente locale. Per visualizzare i modelli disponibili, utilizzate il comando list.

  È necessario specificare `--repository` o `--language` e`--template`.

- `-n`, `--name` — (Facoltativo) Il nome della cartella locale in cui la CLI di GDK inizializza il componente. Specificate una cartella che non esiste. La CLI GDK crea la cartella per te.

  Questa funzionalità è disponibile per GDK CLI v1.1.0 e versioni successive.

Argomenti (inizializzati dal componente della community)

- `-r`, `--repository` — Il componente della community da archiviare nella cartella locale. Per visualizzare i componenti della community disponibili, utilizzate il comando list.

  È necessario specificare `--repository` o `--language` e`--template`.

- `-n`, `--name` — (Facoltativo) Il nome della cartella locale in cui la CLI di GDK inizializza il componente. Specificate una cartella che non esiste. La CLI GDK crea la cartella per te.

  Questa funzionalità è disponibile per GDK CLI v1.1.0 e versioni successive.

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando per inizializzare una cartella di componenti dal modello Python Hello World.

```
$ gdk component init -l python -t HelloWorld
[2021-11-29 12:51:40] INFO - Initializing the project directory with a python
 component template - 'HelloWorld'.
[2021-11-29 12:51:40] INFO - Fetching the component template 'HelloWorld-python'
 from Greengrass Software Catalog.
```

L'esempio seguente mostra l'output prodotto quando si esegue questo comando per inizializzare una cartella di componenti da un componente della comunità.

```
$ gdk component init -r aws-greengrass-labs-database-influxdb
[2022-01-24 15:44:33] INFO - Initializing the project directory with a component
 from repository catalog - 'aws-greengrass-labs-database-influxdb'.
[2022-01-24 15:44:33] INFO - Fetching the component repository 'aws-greengrass-labs-
database-influxdb' from Greengrass Software Catalog.
```

build

Crea il codice sorgente di un componente in una ricetta e in artefatti da pubblicare sul servizio. AWS IoT Greengrass La CLI GDK esegue il sistema di compilazione specificato nel file di configurazione della CLI di GDK,. `gdk-config.json` È necessario eseguire questo comando nella stessa cartella in cui si trova il file. `gdk-config.json`

Quando esegui questo comando, la CLI GDK crea una ricetta e degli artefatti nella cartella `greengrass-build` della cartella dei componenti. La CLI GDK salva la ricetta nella cartella e salva `greengrass-build/recipes` gli artefatti nella cartella. `greengrass-build/artifacts/componentName/componentVersion`

Se utilizzi GDK CLI v1.1.0 o versione successiva, la ricetta del componente può specificare artefatti che esistono in un bucket S3 ma non nella cartella di build del componente locale. È possibile utilizzare questa funzionalità per ridurre l'utilizzo della larghezza di banda quando si sviluppano componenti con artefatti di grandi dimensioni, come i modelli di machine learning.

Dopo aver creato un componente, puoi eseguire una delle seguenti operazioni per testarlo su un dispositivo centrale Greengrass:

- Se sviluppi su un dispositivo diverso da quello su cui esegui il software AWS IoT Greengrass Core, devi pubblicare il componente per distribuirlo su un dispositivo principale Greengrass. Pubblica il componente sul AWS IoT Greengrass servizio e distribuiscilo sul dispositivo principale Greengrass. Per ulteriori informazioni, vedete il comando publish e. Creare distribuzione

- Se sviluppate sullo stesso dispositivo su cui eseguite il software AWS IoT Greengrass Core, potete pubblicare il componente sul AWS IoT Greengrass servizio da distribuire oppure potete creare una distribuzione locale per installare ed eseguire il componente. Per creare una distribuzione locale, usa la CLI di Greengrass. Per ulteriori informazioni, consultare Interfaccia a riga di comando Greengrass e Testare AWS IoT Greengrass i componenti con distribuzioni locali. Quando crei la distribuzione locale, specifica `greengrass-build/recipes` come cartella recipes e `greengrass-build/artifacts` come cartella artifacts.

Riepilogo

```
$ gdk component build
```

Arguments (Argomenti)

Nessuno

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando.

```
$ gdk component build
[2021-11-29 13:18:49] INFO - Getting project configuration from gdk-config.json
[2021-11-29 13:18:49] INFO - Found component recipe file 'recipe.yaml' in the
 project directory.
[2021-11-29 13:18:49] INFO - Building the component 'com.example.PythonHelloWorld'
 with the given project configuration.
[2021-11-29 13:18:49] INFO - Using 'zip' build system to build the component.
[2021-11-29 13:18:49] WARNING - This component is identified as using 'zip' build
 system. If this is incorrect, please exit and specify custom build command in the
 'gdk-config.json'.
[2021-11-29 13:18:49] INFO - Zipping source code files of the component.
[2021-11-29 13:18:49] INFO - Copying over the build artifacts to the greengrass
 component artifacts build folder.
[2021-11-29 13:18:49] INFO - Updating artifact URIs in the recipe.
[2021-11-29 13:18:49] INFO - Creating component recipe in 'C:\Users\MyUser\Documents
\greengrass-components\python\HelloWorld\greengrass-build\recipes'.
```

pubblicazione

Pubblica questo componente nel AWS IoT Greengrass servizio. Questo comando carica gli elementi della build in un bucket S3, aggiorna l'URI degli artefatti nella ricetta e crea una nuova versione

del componente dalla ricetta. L'interfaccia a riga di comando di GDK utilizza il bucket S3 AWS e la regione specificati nel file di configurazione della CLI di GDK,. `gdk-config.json` È necessario eseguire questo comando nella stessa cartella in cui si trova il file. `gdk-config.json`

Se utilizzi GDK CLI v1.1.0 o versione successiva, puoi specificare `--bucket` l'argomento per specificare il bucket S3 in cui la CLI di GDK carica gli artefatti del componente. *Se non specifichi questo argomento, la CLI di GDK viene caricata nel bucket S3 il cui nome bucket-region-accountId è, dove bucket e region sono i valori specificati e AccountID è il tuo IDgdk-config.json.* Account AWS La CLI GDK crea il bucket se non esiste.

Se si utilizza GDK CLI v1.2.0 o versione successiva, è possibile sovrascrivere Regione AWS quanto specificato nel file di configurazione della CLI di GDK utilizzando il parametro. `--region` È inoltre possibile specificare opzioni aggiuntive utilizzando il parametro. `--options` Per un elenco delle opzioni disponibili, vedereFile di configurazione CLI del Greengrass Development Kit.

Quando esegui questo comando, la CLI di GDK pubblica il componente con la versione specificata nella ricetta. Se lo specifichi`NEXT_PATCH`, la CLI di GDK utilizza la versione della patch successiva che non esiste già. Le versioni semantiche utilizzano una major. minore. sistema di numerazione delle patch. Per ulteriori informazioni, consulta la specifica della versione semantica.

> **Note**
>
> Se usi GDK CLI v1.1.0 o versione successiva, quando esegui questo comando, la CLI di GDK verifica se il componente è stato creato. Se il componente non è stato creato, la CLI di GDK crea il componente prima di pubblicarlo.

Riepilogo

```
$ gdk component publish
    [--bucket] [--region] [--options]
```

Arguments (Argomenti)

- `-b`, `--bucket` — (Facoltativo) Specificate il nome del bucket S3 in cui la CLI di GDK pubblica gli artefatti dei componenti.

  Se non specifichi questo argomento, la CLI di GDK viene caricata nel bucket S3 il cui nome bucket-region-accountId è, dove bucket e region

sono i valori specificati e AccountID è il tuo IDgdk-config.json.
Account AWS La CLI GDK crea il bucket se non esiste.

La CLI GDK crea il bucket se non esiste.

Questa funzionalità è disponibile per GDK CLI v1.1.0 e versioni successive.

- `-r`, `--region` — (Facoltativo) Specificate il nome del Regione AWS to al momento della creazione del componente. Questo argomento sovrascrive il nome della regione nella configurazione della CLI di GDK.

  Questa funzionalità è disponibile per GDK CLI v1.2.0 e versioni successive.

- `-o`, `--options` (Facoltativo) Specificate un elenco di opzioni per la pubblicazione di un componente. L'argomento deve essere una stringa JSON valida o un percorso di file a un file JSON contenente le opzioni di pubblicazione. Questo argomento sovrascrive le opzioni nella configurazione della CLI di GDK.

  Questa funzionalità è disponibile per GDK CLI v1.2.0 e versioni successive.

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando.

```
$ gdk component publish
[2021-11-29 13:45:29] INFO - Getting project configuration from gdk-config.json
[2021-11-29 13:45:29] INFO - Found component recipe file 'recipe.yaml' in the
 project directory.
[2021-11-29 13:45:29] INFO - Found credentials in shared credentials file: ~/.aws/
credentials
[2021-11-29 13:45:30] INFO - Publishing the component 'com.example.PythonHelloWorld'
 with the given project configuration.
[2021-11-29 13:45:30] INFO - No private version of the component
 'com.example.PythonHelloWorld' exist in the account. Using '1.0.0' as the next
 version to create.
[2021-11-29 13:45:30] INFO - Uploading the component built artifacts to s3 bucket.
[2021-11-29 13:45:30] INFO - Uploading component artifacts to S3 bucket: {bucket}.
 If this is your first time using this bucket, add the 's3:GetObject' permission
 to each core device's token exchange role to allow it to download the component
 artifacts. For more information, see https://docs.aws.amazon.com/greengrass/v2/
developerguide/device-service-role.html.
[2021-11-29 13:45:30] INFO - Not creating an artifacts bucket as it already exists.
[2021-11-29 13:45:30] INFO - Updating the component recipe
 com.example.PythonHelloWorld-1.0.0.
```

```
[2021-11-29 13:45:30] INFO - Creating a new greengrass component
 com.example.PythonHelloWorld-1.0.0
[2021-11-29 13:45:30] INFO - Created private version '1.0.0' of the component in the
 account.'com.example.PythonHelloWorld'.
```

elenco

Recupera l'elenco dei modelli di componenti e dei componenti della community disponibili.

La CLI GDK recupera i componenti della community dal Greengrass Software Catalog e i modelli di componenti dal AWS IoT Greengrass repository Component Templates in poi. GitHub

È possibile passare l'output di questo comando al comando init per inizializzare i repository di componenti dai modelli e dai componenti della community.

Riepilogo

```
$ gdk component list
    [--template]
    [--repository]
```

Arguments (Argomenti)

- `-t`, `--template` — (Facoltativo) Specificate questo argomento per elencare i modelli di componenti disponibili. Questo comando restituisce il nome e la lingua di ogni modello nel formato*`name`*`-`*`language`*. Ad esempio, in`HelloWorld-python`, il nome del modello è `HelloWorld` e la lingua è`python`.
- `-r`, `--repository` — (Facoltativo) Specificate questo argomento per elencare gli archivi di componenti della community disponibili.

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando.

```
$ gdk component list --template
[2021-11-29 12:29:04] INFO - Listing all the available component templates from
 Greengrass Software Catalog.
[2021-11-29 12:29:04] INFO - Found '2' component templates to display.
1. HelloWorld-python
2. HelloWorld-java
```

config

Usa il `config` comando nell'interfaccia a riga di comando del AWS IoT Greengrass Development Kit (GDK CLI) per modificare la configurazione per il GDK nel file di configurazione,. `gdk-config.json`

Sottocomandi

- update

update

Avvia un prompt interattivo per modificare i campi all'interno di un file di configurazione GDK esistente.

Riepilogo

```
$ gdk config update
    [--component]
```

Arguments (Argomenti)

- `-c`, `--component` — Per aggiornare i campi relativi ai componenti nel file. `gdk-config.json` Questo argomento è obbligatorio in quanto è l'unica opzione.

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando per configurare un componente.

```
$ gdk config update --component
Current value of the REQUIRED component_name is (default:
 com.example.PythonHelloWorld):
Current value of the REQUIRED author is (default: author):
Current value of the REQUIRED version is (default: NEXT_PATCH):
Do you want to change the build configurations? (y/n)
Do you want to change the publish configurations? (y/n)
[2023-09-26 10:19:48] INFO - Config file has been updated. Exiting...
```

test-e2e

Usa il `test-e2e` comando nell'interfaccia a riga di comando del AWS IoT Greengrass Development Kit (GDK CLI) per inizializzare, creare ed end-to-end eseguire moduli di test nel progetto GDK.

Sottocomandi

- init
- build
- run

init

Inizializza un progetto GDK CLI esistente con un modulo di test che utilizza Greengrass Testing Framework (GTF).

Per impostazione predefinita, la CLI GDK recupera il modello del modulo Maven dal AWS IoT Greengrass repository Component Templates in poi. GitHub Questo modulo Maven ha una dipendenza dal file JAR. `aws-greengrass-testing-standalone`

Questo comando crea una nuova directory chiamata `gg-e2e-tests` all'interno del progetto GDK. Se la directory del modulo di test esiste già e non è vuota, il comando esce senza fare nulla. Questa `gg-e2e-tests` cartella contiene la funzionalità Cucumber e le definizioni dei passaggi strutturate in un progetto Maven.

Per impostazione predefinita, questo comando proverà a utilizzare l'ultima versione di GTF.

Riepilogo

```
$ gdk test-e2e init
    [--gtf-version]
```

Arguments (Argomenti)

- `-ov`, `--gtf-version` — (Facoltativo) La versione di GTF da usare con il modulo di end-to-end test nel progetto GDK. Questo valore deve essere una delle versioni GTF delle release. Questo argomento ha la precedenza sulla configurazione della `gtf_version` CLI di GDK.

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando per inizializzare il progetto GDK con il modulo di test.

```
$ gdk test-e2e init
[2023-12-06 12:20:28] INFO - Using the GTF version provided in the GDK test config
 1.2.0
```

```
[2023-12-06 12:20:28] INFO - Downloading the E2E testing template from GitHub into
 gg-e2e-tests directory...
```

build

> **Note**
>
> È necessario creare il componente eseguendolo gdk component build prima di creare il modulo di end-to-end test.

Crea il modulo end-to-end di test. La CLI GDK crea il modulo di test utilizzando il sistema di compilazione specificato nel file di configurazione della CLI GDK, sotto la proprietà. `gdk-config.json` `test-e2e` È necessario eseguire questo comando nella stessa cartella in cui si trova il file. `gdk-config.json`

Per impostazione predefinita, la CLI GDK utilizza il sistema di compilazione maven per creare il modulo di test. Maven è necessario per eseguire il comando. `gdk test-e2e build`

È necessario creare il componente eseguendolo gdk-component-build prima di creare il modulo di test, se i file delle funzionalità di test hanno variabili simili `GDK_COMPONENT_RECIPE_FILE` a `GDK_COMPONENT_NAME` e da interpolare.

Quando esegui questo comando, la CLI GDK interpola tutte le variabili dalla configurazione del progetto GDK e crea il modulo per generare il file JAR di `gg-e2e-tests` test finale.

Riepilogo

```
$ gdk test-e2e build
```

Arguments (Argomenti)

Nessuno

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando.

```
$ gdk test-e2e build
```

```
[2023-07-20 15:36:48] INFO - Updating feature file: file:///path/to//
HelloWorld/greengrass-build/gg-e2e-tests/src/main/resources/greengrass/features/
component.feature
[2023-07-20 15:36:48] INFO - Creating the E2E testing recipe file:///path/to/
HelloWorld/greengrass-build/recipes/e2e_test_recipe.yaml
[2023-07-20 15:36:48] INFO - Building the E2E testing module
[2023-07-20 15:36:48] INFO - Running the build command 'mvn package'
.........
```

run

Esegui il modulo di test con le opzioni di test nel file di configurazione GDK.

> **Note**
>
> È necessario creare il modulo di test eseguendolo gdk test-e2e build prima di eseguire i end-to-end test.

Riepilogo

```
$ gdk test-e2e run
    [--gtf-options]
```

Arguments (Argomenti)

- `-oo`, `--gtf-options` — (Facoltativo) Specificate un elenco di opzioni per l'esecuzione end-to-end dei test. L'argomento deve essere una stringa JSON valida o un percorso di file a un file JSON contenente le opzioni GTF. Le opzioni fornite nel file di configurazione vengono unite a quelle fornite negli argomenti del comando. Se un'opzione è presente in entrambi i posti, quella contenuta nell'argomento ha la precedenza su quella del file di configurazione.

  Se l'`tags`opzione non è specificata in questo comando, GDK la usa per i tag. `Sample` Se non `ggc-archive` è specificato, GDK scarica l'ultima versione dell'archivio Greengrass nucleus.

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando.

```
$ gdk test-e2e run
```

```
[2023-07-20 16:35:53] INFO - Downloading latest nucleus archive from url https://
d2s8p88vqu9w66.cloudfront.net/releases/greengrass-latest.zip
[2023-07-20 16:35:57] INFO - Running test jar with command java -jar /path/to/
greengrass-build/gg-e2e-tests/target/uat-features-1.0.0.jar –ggc-archive=/path/to/
aws-greengrass-gdk-cli/HelloWorld/greengrass-build/greengrass-nucleus-latest.zip –
tags=Sample

16:35:59.693 [] [] [] [INFO]
 com.aws.greengrass.testing.modules.GreengrassContextModule - Extracting /path/
to/workplace/aws-greengrass-gdk-cli/HelloWorld/greengrass-build/greengrass-
nucleus-latest.zip into /var/folders/7g/ltzcb_3s77nbtmkzfb6brwv40000gr/T/gg-
testing-7718418114158172636/greengrass
16:36:00.534 [gtf-1.1.0-SNAPSHOT] [] [] [INFO]
 com.aws.greengrass.testing.features.LoggerSteps - GTF Version is gtf-1.1.0-SNAPSHOT
.......
```

File di configurazione CLI del Greengrass Development Kit

L'interfaccia a riga di comando del AWS IoT Greengrass Development Kit (GDK CLI) legge da un file di configurazione `gdk-config.json` denominato per creare e pubblicare componenti. Questo file di configurazione deve essere presente nella radice del repository dei componenti. Puoi usare il comando init della CLI GDK per inizializzare i repository dei componenti con questo file di configurazione.

Argomenti

- Formato del file di configurazione GDK CLI
- Esempi di file di configurazione CLI GDK

Formato del file di configurazione GDK CLI

Quando definisci un file di configurazione GDK CLI per un componente, specifichi le seguenti informazioni in formato JSON.

`gdk_version`

La versione minima della CLI GDK compatibile con questo componente. Questo valore deve essere una delle versioni della CLI di GDK delle release.

`component`

La configurazione per questo componente.

componentName

`author`

L'autore o l'editore del componente.

`version`

La versione del componente. Specifica una delle seguenti proprietà:

- `NEXT_PATCH`— Quando scegliete questa opzione, la CLI di GDK imposta la versione quando pubblicate il componente. La CLI GDK interroga AWS IoT Greengrass il servizio per identificare l'ultima versione pubblicata del componente. Quindi, imposta la versione alla versione della patch successiva a quella versione. Se non hai mai pubblicato il componente prima, la CLI di GDK utilizza la versione. `1.0.0`

  Se scegli questa opzione, non puoi utilizzare la CLI di Greengrass per distribuire e testare localmente il componente sul tuo computer di sviluppo locale che esegue il software Core. AWS IoT Greengrass Per abilitare le distribuzioni locali, devi invece specificare una versione semantica.
- Una versione semantica, ad esempio. **`1.0.0`** Le versioni semantiche utilizzano un major. minore. sistema di numerazione delle patch. Per ulteriori informazioni, consulta la specifica della versione semantica.

  Se sviluppate componenti su un dispositivo Greengrass core su cui desiderate distribuire e testare il componente, scegliete questa opzione. È necessario creare il componente con una versione specifica per creare distribuzioni locali con la Greengrass CLI.

`build`

La configurazione da utilizzare per creare il codice sorgente di questo componente in artefatti. Questo oggetto contiene le seguenti informazioni:

`build_system`

Il sistema di compilazione da utilizzare. Seleziona una delle opzioni seguenti:

- `zip`— Impacchetta la cartella del componente in un file ZIP da definire come unico elemento del componente. Scegliete questa opzione per i seguenti tipi di componenti:
  - Componenti che utilizzano linguaggi di programmazione interpretati, come Python o. JavaScript
  - Componenti che impacchettano file diversi dal codice, come modelli di apprendimento automatico o altre risorse.

La CLI GDK comprime la cartella del componente in un file zip con lo stesso nome della cartella del componente. Ad esempio, se il nome della cartella del componente è`HelloWorld`, la CLI GDK crea un file zip denominato. `HelloWorld.zip`

> **Note**
>
> Se si utilizza la versione 1.0.0 della CLI di GDK su un dispositivo Windows, i nomi delle cartelle dei componenti e dei file zip devono contenere solo lettere minuscole.

Quando la CLI di GDK comprime la cartella del componente in un file zip, salta i seguenti file:

- Il file `gdk-config.json`
- Il file della ricetta (o) `recipe.json` `recipe.yaml`
- Crea cartelle, come `greengrass-build`

- `maven`— Esegue il `mvn clean package` comando per creare il codice sorgente del componente in artefatti. Scegli questa opzione per i componenti che utilizzano Maven, come i componenti Java.

  Sui dispositivi Windows, questa funzionalità è disponibile per GDK CLI v1.1.0 e versioni successive.
- `gradle`— Esegue il `gradle build` comando per creare il codice sorgente del componente in artefatti. Scegliete questa opzione per i componenti che utilizzano Gradle. Questa funzionalità è disponibile per GDK CLI v1.1.0 e versioni successive.

  Il sistema di `gradle` build supporta Kotlin DSL come file di build. Questa funzionalità è disponibile per GDK CLI v1.2.0 e versioni successive.
- `gradlew`— Esegue il `gradlew` comando per creare il codice sorgente del componente in artefatti. Scegliete questa opzione per i componenti che utilizzano il Gradle Wrapper.

  Questa funzionalità è disponibile per GDK CLI v1.2.0 e versioni successive.
- `custom`— Esegue un comando personalizzato per creare il codice sorgente del componente in una ricetta e in artefatti. Specificate il comando personalizzato nel `custom_build_command` parametro.

`custom_build_command`

(Facoltativo) Il comando custom build da eseguire per un sistema di build personalizzato. È necessario specificare questo parametro se si specifica `custom` per`build_system`.

> **Important**
>
> Questo comando deve creare una ricetta e degli artefatti nelle seguenti cartelle all'interno della cartella del componente. La CLI GDK crea queste cartelle per te quando esegui il comando component build.
>
> - Cartella delle ricette: `greengrass-build/recipes`
> - Cartella Artifacts: `greengrass-build/artifacts/componentName/componentVersion`
>
>   Sostituisci *componentName* con il nome del componente e sostituisci *componentVersion con la versione* del componente o. `NEXT_PATCH`

È possibile specificare una singola stringa o un elenco di stringhe, in cui ogni stringa è una parola del comando. Ad esempio, per eseguire un comando di compilazione personalizzato per un componente C++, puoi specificare **`cmake --build build --config Release`** o. **`["cmake", "--build", "build", "--config", "Release"]`**

Per visualizzare un esempio di sistema di compilazione personalizzato, vedi aws.greengrass.labs.LocalWebServer community componenton GitHub.

`options`

(Facoltativo) Opzioni di configurazione aggiuntive utilizzate durante il processo di creazione del componente.

Questa funzionalità è disponibile per GDK CLI v1.2.0 e versioni successive.

`excludes`

Un elenco di pattern a glob che definiscono quali file escludere dalla directory dei componenti durante la creazione del file zip. Valido solo quando `build_system` è`zip`.

> **Note**
>
> Nelle versioni 1.4.0 e precedenti della CLI di GDK, qualsiasi file che corrisponde a una voce nell'elenco delle esclusioni viene escluso da tutte le sottodirectory del componente. Per ottenere lo stesso comportamento nelle versioni 1.5.0 e successive della CLI di GDK, `**/` anteponi le voci esistenti nell'elenco delle esclusioni. Ad esempio, `*.txt` escluderà i file di testo solo dalla directory; `**/*.txt` escluderà i file di testo da tutte le directory e sottodirectory.
> Nelle versioni 1.5.0 e successive della CLI di GDK, potresti visualizzare un avviso durante la compilazione del componente `excludes` quando è definito nel file di configurazione GDK. Per disabilitare questo avviso, imposta la variabile di ambiente su. `GDK_EXCLUDES_WARN_IGNORE` `true`

La CLI GDK esclude sempre i seguenti file dal file zip:

- Il file `gdk-config.json`
- Il file della ricetta (o) `recipe.json` `recipe.yaml`
- Crea cartelle, come `greengrass-build`

I seguenti file sono esclusi per impostazione predefinita. Tuttavia, puoi controllare quali di questi file sono esclusi con l'`excludes`opzione.

- Qualsiasi cartella che inizia con il prefisso «test» () `test*`
- Tutti i file nascosti
- La cartella `node_modules`

Se specificate l'`excludes`opzione, la CLI di GDK esclude solo i file impostati con l'opzione. `excludes` Se non specificate l'`excludes`opzione, la CLI di GDK esclude i file e le cartelle predefiniti precedentemente indicati.

`zip_name`

Il nome del file zip da usare quando si crea un elemento zip durante il processo di compilazione. Valido solo quando è. `build_system` `zip` Se `build_system` è vuoto, il nome del componente viene utilizzato per il nome del file zip.

`publish`

La configurazione da utilizzare per pubblicare questo componente nel AWS IoT Greengrass servizio.

Se utilizzi GDK CLI v1.1.0 o versione successiva, puoi specificare `--bucket` l'argomento per specificare il bucket S3 in cui la CLI di GDK carica gli artefatti del componente. *Se non specifichi questo argomento, la CLI di GDK viene caricata nel bucket S3 il cui nome bucket-region-accountId è, dove bucket e region sono i valori specificati e AccountID è il tuo IDgdk-config.json.* Account AWS La CLI GDK crea il bucket se non esiste.

Questo oggetto contiene le seguenti informazioni:

`bucket`

Il nome del bucket S3 da usare per ospitare gli artefatti dei componenti.

`region`

Il Regione AWS punto in cui la CLI di GDK pubblica questo componente.

Questa proprietà è facoltativa se si utilizza GDK CLI v1.3.0 o versione successiva.

`options`

(Facoltativo) Opzioni di configurazione aggiuntive utilizzate durante la creazione della versione del componente.

Questa funzionalità è disponibile per GDK CLI v1.2.0 e versioni successive.

`file_upload_args`

Una struttura JSON contenente argomenti inviati ad Amazon S3 durante il caricamento di file in un bucket, come metadati e meccanismi di crittografia. Per un elenco degli argomenti consentiti, consulta la *S3Transfer*classe nella documentazione di Boto3. .

`test-e2e`

(Facoltativo) La configurazione da utilizzare durante il end-to-end test del componente. Questa funzionalità è disponibile per GDK CLI v1.3.0 e versioni successive.

`build`

`build_system`— Il sistema di compilazione da utilizzare. L'opzione predefinita è`maven`. Seleziona una delle opzioni seguenti:

- `maven`— Esegue il `mvn package` comando per creare il modulo di test. Scegli questa opzione per creare il modulo di test che utilizza Maven.
- `gradle`— Esegue il `gradle build` comando per creare il modulo di test. Scegliete questa opzione per il modulo di test che utilizza Gradle.

`gtf_version`

(Facoltativo) La versione del Greengrass Testing Framework (GTF) da usare come dipendenza del modulo di end-to-end test quando inizializzi il progetto GDK con GTF. Questo valore deve essere una delle versioni GTF delle release. L'impostazione predefinita è GTF versione 1.1.0.

`gtf_options`

(Facoltativo) Opzioni di configurazione aggiuntive utilizzate durante il end-to-end test del componente.

L'elenco seguente include le opzioni che è possibile utilizzare con la versione 1.1.0 di GTF.

- `additional-plugins`— (Opzionale) Plugin aggiuntivi per Cucumber
- `aws-region`— Si rivolge a endpoint regionali specifici per i servizi. AWS L'impostazione predefinita è ciò che rileva l'AWSSDK.
- `credentials-path`— Percorso opzionale AWS delle credenziali del profilo. L'impostazione predefinita sono le credenziali rilevate nell'ambiente host.
- `credentials-path-rotation`— Durata di rotazione opzionale per le credenziali. AWS Il valore predefinito è 15 minuti o. `PT15M`
- `csr-path`— Il percorso della CSR con cui verrà generato il certificato del dispositivo.
- `device-mode`— Il dispositivo bersaglio sottoposto a test. L'impostazione predefinita è il dispositivo locale.
- `env-stage`— Si rivolge all'ambiente di implementazione di Greengrass. L'impostazione predefinita è la produzione.
- `existing-device-cert-arn`— L'arn di un certificato esistente che desideri utilizzare come certificato del dispositivo per Greengrass.
- `feature-path`— File o directory contenente file di funzionalità aggiuntivi. L'impostazione predefinita è che non vengono utilizzati file di feature aggiuntivi.

- `gg-cli-version`— Sostituisce la versione della CLI di Greengrass. Il valore predefinito è il valore trovato in. `ggc.version`
- `gg-component-bucket`— Il nome di un bucket Amazon S3 esistente che contiene componenti Greengrass.
- `gg-component-overrides`— Un elenco di componenti sostituiti da Greengrass.
- `gg-persist`— Un elenco di elementi di test che devono persistere dopo l'esecuzione di un test. Il comportamento predefinito consiste nel non rendere persistente nulla. I valori accettati sono: `aws.resourcesinstalled.software`, `egenerated.files`.
- `gg-runtime`— Un elenco di valori per influenzare il modo in cui il test interagisce con le risorse di test. Questi valori sostituiscono il parametro. `gg.persist` Se l'impostazione predefinita è vuota, presuppone che tutte le risorse di test siano gestite per test case, incluso il runtime Greengrass installato. I valori accettati sono:`aws.resources`, `einstalled.software`. `generated.files`
- `ggc-archive`— Il percorso verso il componente Greengrass nucleus archiviato.
- `ggc-install-root`— Directory per installare il componente Greengrass nucleus. I valori predefiniti sono test.temp.path e test run.
- `ggc-log-level`— Imposta il livello di registro del nucleo di Greengrass per l'esecuzione del test. L'impostazione predefinita è «INFO».
- `ggc-tes-rolename`— Il ruolo IAM che AWS IoT Greengrass Core assumerà per accedere ai AWS servizi. Se non esiste un ruolo con un nome specifico, ne verrà creato uno e una politica di accesso predefinita.
- `ggc-trusted-plugins`— L'elenco separato da virgole dei percorsi (sull'host) dei plugin affidabili che devono essere aggiunti a Greengrass. Per fornire il percorso sul DUT stesso, inserisci il prefisso 'dut: '
- `ggc-user-name`— Il valore PosixUser user:group per il nucleo Greengrass. Il valore predefinito è il nome utente corrente a cui è stato effettuato l'accesso.
- `ggc-version`— Sostituisce la versione del componente Greengrass nucleus in esecuzione. Il valore predefinito è il valore trovato in ggc.archive.
- `log-level`— Livello di registro dell'esecuzione del test. Il valore predefinito è «INFO».
- `parallel-config`— Set di indice di batch e numero di batch come stringa JSON. Il valore predefinito dell'indice batch è 0 e il numero di batch è 1.
- `proxy-url`— Configura tutti i test per indirizzare il traffico attraverso questo URL.
- `tags`— Esegui solo tag di funzionalità. Può essere intersecato con '&'

- `test-id-prefix`— Un prefisso comune applicato a tutte le risorse specifiche del test, inclusi i nomi e i tag AWS delle risorse. L'impostazione predefinita è il prefisso «gg».
- `test-log-path`— Directory che conterrà i risultati dell'intera esecuzione del test. Il valore predefinito è «testResults».
- `test-results-json`— Contrassegno per determinare se viene generato un report Cucumber JSON risultante scritto su disco. Il valore predefinito è true.
- `test-results-log`— Contrassegno per determinare se l'output della console viene generato scritto su disco. Il valore predefinito è false (falso).
- `test-results-xml`— Contrassegno per determinare se un report XML JUnit risultante viene generato e scritto su disco. Il valore predefinito è true.
- `test-temp-path`— Directory per generare artefatti di test locali. Il valore predefinito è una directory temporanea casuale con il prefisso gg-testing.
- `timeout-multiplier`— Moltiplicatore fornito a tutti i timeout dei test. Il valore predefinito è 1.0.

Esempi di file di configurazione CLI GDK

Puoi fare riferimento ai seguenti esempi di file di configurazione della CLI GDK per aiutarti a configurare gli ambienti dei componenti Greengrass.

Ciao mondo (Python)

Il seguente file di configurazione GDK CLI supporta un componente Hello World che esegue uno script Python. Questo file di configurazione utilizza il sistema di `zip` compilazione per impacchettare lo script Python del componente in un file ZIP che la CLI di GDK carica come artefatto.

```
{
  "component": {
    "com.example.PythonHelloWorld": {
      "author": "Amazon",
      "version": "NEXT_PATCH",
      "build": {
        "build_system" : "zip",
        "options": {
           "excludes": [".*"]
        }
      },
      "publish": {
        "bucket": "greengrass-component-artifacts",
```

```
        "region": "us-west-2",
        "options": {
           "file_upload_args": {
              "Metadata": {
                 "some-key": "some-value"
              }
           }
        }
      }
    },
  "test-e2e":{
    "build":{
        "build_system": "maven"
    },
    "gtf_version": "1.1.0",
    "gtf_options": {
         "tags": "Sample"
     }
  },
  "gdk_version": "1.6.1"
  }
}
```

Hello World (Java)

Il seguente file di configurazione GDK CLI supporta un componente Hello World che esegue un'applicazione Java. Questo file di configurazione utilizza il sistema di `maven` compilazione per impacchettare il codice sorgente Java del componente in un file JAR che la CLI GDK carica come artefatto.

```
{
  "component": {
    "com.example.JavaHelloWorld": {
      "author": "Amazon",
      "version": "NEXT_PATCH",
      "build": {
        "build_system" : "maven"
      },
      "publish": {
        "bucket": "greengrass-component-artifacts",
        "region": "us-west-2",
        "options": {
           "file_upload_args": {
```

```
            "Metadata": {
               "some-key": "some-value"
            }
         }
      }
   }
  },
  "test-e2e":{
    "build":{
        "build_system": "maven"
    },
    "gtf_version": "1.1.0",
    "gtf_options": {
         "tags": "Sample"
     }
  },
  "gdk_version": "1.6.1"
  }
}
```

Componenti della community

Diversi componenti della community nel Greengrass Software Catalog utilizzano la CLI GDK. Puoi esplorare i file di configurazione della CLI di GDK nei repository di questi componenti.

Per visualizzare i file di configurazione della CLI GDK dei componenti della community

1. Esegui il comando seguente per elencare i componenti della community che utilizzano la CLI GDK.

   ```
   gdk component list --repository
   ```

   La risposta elenca il nome del GitHub repository per ogni componente della comunità che utilizza la CLI GDK. Ogni repository esiste nell'organizzazione. `awslabs`

   ```
   [2022-02-22 17:27:31] INFO - Listing all the available component repositories from
    Greengrass Software Catalog.
   [2022-02-22 17:27:31] INFO - Found '6' component repositories to display.
   1. aws-greengrass-labs-database-influxdb
   2. aws-greengrass-labs-telemetry-influxdbpublisher
   3. aws-greengrass-labs-dashboard-grafana
   4. aws-greengrass-labs-dashboard-influxdb-grafana
   ```

```
5. aws-greengrass-labs-local-web-server
6. aws-greengrass-labs-lookoutvision-gstreamer
```

2. Apri l' GitHub archivio di un componente della community al seguente URL. Sostituiscilo *`community-component-name`*con il nome di un componente della community del passaggio precedente.

   ```
   https://github.com/awslabs/community-component-name
   ```

Interfaccia a riga di comando Greengrass

L'interfaccia a riga di comando (CLI) di Greengrass consente di interagire con AWS IoT Greengrass Core sul dispositivo per sviluppare componenti ed eseguire il debug a livello locale. Ad esempio, è possibile utilizzare la CLI Greengrass per creare una distribuzione locale e riavviare un componente sul dispositivo principale.

Implementa il `aws.greengrass.Cli` componente Greengrass CLI () per installare la Greengrass CLI sul tuo dispositivo principale.

> **Important**
>
> Si consiglia di utilizzare questo componente solo in ambienti di sviluppo, non in ambienti di produzione. Questo componente fornisce l'accesso a informazioni e operazioni che in genere non sono necessarie in un ambiente di produzione. Segui il principio del privilegio minimo distribuendo questo componente solo sui dispositivi principali dove ne hai bisogno.

Argomenti

- Installazione della CLI di Greengrass
- Comandi della CLI di Greengrass

Installazione della CLI di Greengrass

È possibile installare la Greengrass CLI in uno dei seguenti modi:

- Usa l'`--deploy-dev-tools`argomento quando configuri per la prima volta il software AWS IoT Greengrass Core sul tuo dispositivo. È inoltre necessario specificare `--provision true` di applicare questo argomento.

- Implementa il `aws.greengrass.Cli` componente Greengrass CLI () sul tuo dispositivo.

Questa sezione descrive i passaggi per distribuire il componente Greengrass CLI. Per informazioni sull'installazione della Greengrass CLI durante la configurazione iniziale, vedere. Tutorial: Nozioni di base su AWS IoT Greengrass V2

Prerequisiti

Per distribuire il componente Greengrass CLI, è necessario soddisfare i seguenti requisiti:

- AWS IoT Greengrass Software di base installato e configurato sul dispositivo principale. Per ulteriori informazioni, consulta Tutorial: Nozioni di base su AWS IoT Greengrass V2.
- Per utilizzare AWS CLI per distribuire la CLI Greengrass, è necessario aver installato e configurato il. AWS CLI Per ulteriori informazioni, consulta Configurazione della AWS CLI nella Guida per l'utente di AWS Command Line Interface .
- È necessario essere autorizzati a utilizzare la CLI Greengrass per interagire con il software Core. AWS IoT Greengrass Effettuate una delle seguenti operazioni per utilizzare la Greengrass CLI:
  - Utilizzate l'utente di sistema che esegue il software AWS IoT Greengrass Core.
  - Usa un utente con autorizzazioni root o amministrative. Sui dispositivi principali di Linux, puoi utilizzarlo per ottenere i permessi `sudo` di root.
  - Usa un utente di sistema che fa parte di un gruppo specificato nei parametri di `AuthorizedWindowsGroups` configurazione `AuthorizedPosixGroups` o quando distribuisci il componente. Per ulteriori informazioni, consulta Configurazione dei componenti della CLI di Greengrass.

Implementa il componente Greengrass CLI

Completa i seguenti passaggi per distribuire il componente Greengrass CLI sul tuo dispositivo principale:

Per distribuire il componente Greengrass CLI (console)

1. Accedi alla console AWS IoT Greengrass.
2. Nel menu di navigazione, scegli Componenti.
3. Nella pagina Componenti, nella scheda Componenti pubblici, scegli `aws.greengrass.Cli`.
4. Nella pagina aws.greengrass.Cli, scegli (Distribuisci).
5. Da Aggiungi alla distribuzione, scegli Crea nuova distribuzione.

6. Nella pagina Specificare la destinazione, in Destinazioni di distribuzione, nell'elenco Nome destinazione, scegli il gruppo Greengrass in cui desideri eseguire la distribuzione e scegli Avanti.
7. Nella pagina Seleziona componenti, verifica che il aws.greengrass.Clicomponente sia selezionato e scegli Avanti.
8. Nella pagina Configura componenti, mantieni le impostazioni di configurazione predefinite e scegli Avanti.
9. Nella pagina Configura impostazioni avanzate, mantieni le impostazioni di configurazione predefinite e scegli Avanti.
10. Nella pagina di revisione, fai clic su Distribuisci

Per distribuire il componente Greengrass CLI ()AWS CLI

1. Sul tuo dispositivo, crea un `deployment.json` file per definire la configurazione di distribuzione per il componente Greengrass CLI. Questo file dovrebbe avere il seguente aspetto:

```
{
  "targetArn":"targetArn",
  "components": {
    "aws.greengrass.Cli": {
      "componentVersion": "2.12.3",
      "configurationUpdate": {
        "merge": "{\"AuthorizedPosixGroups\":\"<group1>,<group2>,...,<groupN>\",
\"AuthorizedWindowsGroups\":\"<group1>,<group2>,...,<groupN>\"}"
      }
    }
  }
}
```

   - Nel campo `target`, sostituisci *`targetArn`* con il nome della risorsa Amazon (ARN) dell'oggetto o del gruppo di oggetti a cui destinare la distribuzione, nel seguente formato:
     - Oggetto: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
     - Gruppo di oggetti: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*
   - Nell'oggetto `aws.greengrass.Cli` componente, specificate i valori come segue:

     `version`

     La versione del componente Greengrass CLI.

`configurationUpdate.AuthorizedPosixGroups`

(Facoltativo) Una stringa che contiene un elenco di gruppi di sistema separati da virgole. Autorizzi questi gruppi di sistema a utilizzare la CLI Greengrass per interagire con AWS IoT Greengrass il software Core. È possibile specificare nomi o ID di gruppo. Ad esempio, `group1,1002,group3` autorizza tre gruppi di sistema (`group1`1002, e`group3`) a utilizzare la CLI Greengrass.

Se non specifichi alcun gruppo da autorizzare, puoi utilizzare la CLI Greengrass come `sudo` utente root () o come utente di sistema che AWS IoT Greengrass esegue il software Core.

`configurationUpdate.AuthorizedWindowsGroups`

(Facoltativo) Una stringa che contiene un elenco separato da virgole di gruppi di sistema. Autorizzi questi gruppi di sistema a utilizzare la CLI Greengrass per interagire con AWS IoT Greengrass il software Core. È possibile specificare nomi o ID di gruppo. Ad esempio, `group1,1002,group3` autorizza tre gruppi di sistema (`group1`1002, e`group3`) a utilizzare la CLI Greengrass.

Se non specifichi alcun gruppo da autorizzare, puoi utilizzare la CLI Greengrass come amministratore o come utente di sistema che AWS IoT Greengrass esegue il software Core.

2. Esegui il comando seguente per distribuire il componente Greengrass CLI sul dispositivo:

```
$ aws greengrassv2 create-deployment --cli-input-json file://path/
to/deployment.json
```

Durante l'installazione, il componente aggiunge un collegamento simbolico */greengrass/v2*/bin nella cartella del dispositivo e da questo percorso si esegue la Greengrass CLI. `greengrass-cli` Per eseguire la CLI di Greengrass senza il percorso assoluto, aggiungi la */greengrass/v2*/bin cartella alla variabile PATH. Per verificare l'installazione della CLI di Greengrass, esegui il seguente comando:

Linux or Unix

```
/greengrass/v2/bin/greengrass-cli help
```

Windows

```
C:\greengrass\v2\bin\greengrass-cli help
```

Verrà visualizzato l'output seguente:

```
Usage: greengrass-cli [-hV] [--ggcRootPath=<ggcRootPath>] [COMMAND]
Greengrass command line interface

      --ggcRootPath=<ggcRootPath>
                  The AWS IoT Greengrass V2 root directory.
  -h, --help      Show this help message and exit.
  -V, --version   Print version information and exit.
Commands:
  help                Show help information for a command.
  component           Retrieve component information and stop or restart
                        components.
  deployment          Create local deployments and retrieve deployment status.
  logs                Analyze Greengrass logs.
  get-debug-password  Generate a password for use with the HTTP debug view
                        component.
```

Se `greengrass-cli` non viene trovato, la distribuzione potrebbe non essere riuscita a installare la CLI di Greengrass. Per ulteriori informazioni, consulta Risoluzione dei problemi AWS IoT Greengrass V2.

Comandi della CLI di Greengrass

La CLI di Greengrass fornisce un'interfaccia a riga di comando per interagire localmente con il dispositivo principale. AWS IoT Greengrass I comandi della CLI di Greengrass utilizzano il seguente formato.

```
$ greengrass-cli <command> <subcommand> [arguments]
```

Per impostazione predefinita, il file `greengrass-cli` eseguibile contenuto nella */greengrass/v2*`/bin/` cartella interagisce con la versione del software AWS IoT Greengrass Core in esecuzione nella cartella. */greengrass/v2* Se chiamate un file eseguibile che non si trova in questa posizione o se desiderate interagire con il software AWS IoT Greengrass Core in una posizione diversa, dovete utilizzare uno dei seguenti metodi per specificare in modo esplicito il percorso principale del software AWS IoT Greengrass Core con cui desiderate interagire:

- Impostare la variabile di ambiente `GGC_ROOT_PATH` su */greengrass/v2*.
- Aggiungete l'`--ggcRootPath` */greengrass/v2*argomento al comando come illustrato nell'esempio seguente.

```
greengrass-cli --ggcRootPath /greengrass/v2 <command> <subcommand> [arguments]
```

È possibile utilizzare i seguenti argomenti con qualsiasi comando:

- Utilizzare `--help` per informazioni su uno specifico comando della CLI di Greengrass.
- Utilizzare `--version` per informazioni sulla versione della CLI di Greengrass.

Questa sezione descrive i comandi della CLI di Greengrass e fornisce esempi per questi comandi. La sinossi di ogni comando mostra i relativi argomenti e il loro utilizzo. Gli argomenti opzionali sono indicati tra parentesi quadre.

Comandi disponibili

- componente
- implementazione
- log
- get-debug-password

componente

Usa il `component` comando per interagire con i componenti locali sul tuo dispositivo principale.

Sottocomandi

- details
- list
- riavviare
- arresta

details

Recupera la versione, lo stato e la configurazione di un componente.

Riepilogo

```
greengrass-cli component details --name <component-name>
```

Arguments (Argomenti)

`--name,-n`. Il nome del componente.

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando.

```
$ sudo greengrass-cli component details --name MyComponent

Component Name: MyComponent
Version: 1.0.0
State: RUNNING
Configuration: null
```

list

Recupera il nome, la versione, lo stato e la configurazione di ogni componente installato sul dispositivo.

Riepilogo

```
greengrass-cli component list
```

Arguments (Argomenti)

Nessuno

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando.

```
$ sudo greengrass-cli component list

Components currently running in Greengrass:
Component Name: FleetStatusService
Version: 0.0.0
State: RUNNING
Configuration: {"periodicUpdateIntervalSec":86400.0}
```

```
Component Name: UpdateSystemPolicyService
Version: 0.0.0
State: RUNNING
Configuration: null
Component Name: aws.greengrass.Nucleus
Version: 2.0.0
State: FINISHED
Configuration: {"awsRegion":"region","runWithDefault":
{"posixUser":"ggc_user:ggc_group"},"telemetry":{}}
Component Name: DeploymentService
Version: 0.0.0
State: RUNNING
Configuration: null
Component Name: TelemetryAgent
Version: 0.0.0
State: RUNNING
Configuration: null
Component Name: aws.greengrass.Cli
Version: 2.0.0
State: RUNNING
Configuration: {"AuthorizedPosixGroups":"ggc_user"}
```

riavviare

Riavvia i componenti.

Riepilogo

```
greengrass-cli component restart --names <component-name>,...
```

Arguments (Argomenti)

`--names,-n`. Il nome del componente. È richiesto almeno un nome di componente. È possibile specificare nomi di componenti aggiuntivi, separando ogni nome con una virgola.

Output

Nessuno

arresta

Smettete di far funzionare i componenti.

Riepilogo

```
greengrass-cli component stop --names <component-name>,...
```

Arguments (Argomenti)

`--names,-n`. Il nome del componente. È richiesto almeno un nome di componente. Se necessario, è possibile specificare nomi di componenti aggiuntivi, separando ogni nome con una virgola.

Output

Nessuno

implementazione

Utilizzate il `deployment` comando per interagire con i componenti locali del dispositivo principale.

Per monitorare l'avanzamento di una distribuzione locale, usa il `status` sottocomando. Non è possibile monitorare l'avanzamento di una distribuzione locale utilizzando la console.

Sottocomandi

- Crea
- annullare
- elenco
- status

Crea

Crea o aggiorna una distribuzione locale utilizzando ricette di componenti, artefatti e argomenti di runtime specifici.

Riepilogo

```
greengrass-cli deployment create
    --recipeDir path/to/component/recipe
    [--artifactDir path/to/artifact/folder ]
    [--update-config {component-configuration}]
```

```
    [--groupId <thing-group>]
    [--merge "<component-name>=<component-version>"]...
    [--runWith "<component-name>:posixUser=<user-name>[:<group-name>]"]...
    [--systemLimits "{component-system-resource-limits}]"]...
    [--remove <component-name>,...]
    [--failure-handling-policy <policy name[ROLLBACK, DO_NOTHING]>]
```

Arguments (Argomenti)

- `--recipeDir`,. `-r` Il percorso completo della cartella che contiene i file di ricette dei componenti.
- `--artifactDir`,`-a`. Il percorso completo della cartella che contiene i file degli artefatti da includere nella distribuzione. La cartella artifacts deve contenere la seguente struttura di directory:

  ```
  /path/to/artifact/folder/<component-name>/<component-version>/<artifacts>
  ```

- `--update-config`,. `-c` Gli argomenti di configurazione per la distribuzione, forniti come stringa JSON o file JSON. La stringa JSON deve avere il seguente formato:

  ```
  { \
    "componentName": { \
      "MERGE": {"config-key": "config-value"}, \
      "RESET": ["path/to/reset/"] \
    } \
  }
  ```

  `MERGE`e `RESET` fanno distinzione tra maiuscole e minuscole e devono essere scritte in maiuscolo.
- `--groupId`,. `-g` Il gruppo di oggetti bersaglio per la distribuzione.
- `--merge`,`-m`. Il nome e la versione del componente di destinazione che desideri aggiungere o aggiornare. È necessario fornire le informazioni sul componente nel formato*<component>*=*<version>*. Utilizzate un argomento separato per ogni componente aggiuntivo da specificare. Se necessario, utilizzate l'`--runWith`hargomento per fornire le `windowsUser` informazioni `posixUserposixGroup`, e per l'esecuzione del componente.
- `--runWith`. `posixUserposixGroup`, e `windowsUser` informazioni per l'esecuzione di un componente generico o Lambda. È necessario fornire queste informazioni nel formato*<component>*`:{posixUser|windowsUser}`=*<user>*`[:`*<=posixGroup>*`]`. Ad esempio, è possibile specificare **`HelloWorld:posixUser=ggc_user:ggc_group`**

o**`HelloWorld:windowsUser=ggc_user`**. Utilizzate un argomento separato per ogni opzione aggiuntiva da specificare.

Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti.

- `--systemLimits`. I limiti delle risorse di sistema da applicare ai processi dei componenti Lambda generici e non containerizzati sul dispositivo principale. È possibile configurare la quantità massima di utilizzo di CPU e RAM utilizzabile dai processi di ciascun componente. Specificate un oggetto JSON serializzato o il percorso di un file JSON. L'oggetto JSON deve avere il seguente formato.

```
{  \
  "componentName": { \
    "cpus": cpuTimeLimit, \
    "memory": memoryLimitInKb \
  } \
}
```

È possibile configurare i seguenti limiti di risorse di sistema per ogni componente:

- `cpus`— La quantità massima di tempo di CPU che i processi di questo componente possono utilizzare sul dispositivo principale. Il tempo totale della CPU di un dispositivo principale è equivalente al numero di core CPU del dispositivo. Ad esempio, su un dispositivo principale con 4 core CPU, è possibile impostare questo valore in modo da 2 limitare i processi di questo componente al 50% di utilizzo di ciascun core della CPU. Su un dispositivo con 1 core di CPU, puoi impostare questo valore `0.25` per limitare i processi di questo componente al 25 percento di utilizzo della CPU. Se imposti questo valore su un numero maggiore del numero di core della CPU, il software AWS IoT Greengrass Core non limita l'utilizzo della CPU del componente.
- `memory`— La quantità massima di RAM (in kilobyte) che i processi di questo componente possono utilizzare sul dispositivo principale.

Per ulteriori informazioni, consulta Configura i limiti delle risorse di sistema per i componenti.

Questa funzionalità è disponibile per la versione 2.4.0 e successive del componente Greengrass nucleus e della Greengrass CLI sui dispositivi core Linux. AWS IoT Greengrass attualmente non supporta questa funzionalità sui dispositivi Windows core.

- `--remove`. Il nome del componente di destinazione che si desidera rimuovere da una distribuzione locale. Per rimuovere un componente che è stato unito da una distribuzione cloud, è necessario fornire l'ID di gruppo del thing group di destinazione nel seguente formato:

  Greengrass nucleus v2.4.0 and later

  ```
  --remove <component-name> --groupId <group-name>
  ```

  Earlier than v2.4.0

  ```
  --remove <component-name> --groupId thinggroup/<group-name>
  ```

- `--failure-handling-policy`. Definisce l'azione intrapresa in caso di errore di una distribuzione. È possibile specificare due azioni:
  - `ROLLBACK` –
  - `DO_NOTHING` –

  Questa funzionalità è disponibile per la versione 2.11.0 e successive di. Nucleo Greengrass

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando.

```
$ sudo greengrass-cli deployment create \
    --merge MyApp1=1.0.0 \
    --merge MyApp2=1.0.0 --runWith MyApp2:posixUser=ggc_user \
    --remove MyApp3 \
    --recipeDir recipes/ \
    --artifactDir artifacts/

Local deployment has been submitted! Deployment Id: 44d89f46-1a29-4044-
ad89-5151213dfcbc
```

annullare

Annulla la distribuzione specificata.

Riepilogo

```
greengrass-cli deployment cancel
    -i <deployment-id>
```

Argomenti

`-i`. L'identificatore univoco della distribuzione da annullare. L'ID di distribuzione viene restituito nell'output del `create` comando.

Output

- Nessuno

elenco

Recupera lo stato delle ultime 10 distribuzioni locali.

Riepilogo

```
greengrass-cli deployment list
```

Arguments (Argomenti)

Nessuno

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando. A seconda dello stato della distribuzione, l'output mostra uno dei seguenti valori di stato: `IN_PROGRESS`SUCCEEDED, o`FAILED`.

```
$ sudo greengrass-cli deployment list

44d89f46-1a29-4044-ad89-5151213dfcbc: SUCCEEDED
Created on: 6/27/23 11:05 AM
```

status

Recupera lo stato di una distribuzione specifica.

Riepilogo

```
greengrass-cli deployment status -i <deployment-id>
```

Arguments (Argomenti)

`-i`. L'ID della distribuzione.

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando. A seconda dello stato della distribuzione, l'output mostra uno dei seguenti valori di stato: `IN_PROGRESS`, `SUCCEEDED` o `FAILED`.

```
$ sudo greengrass-cli deployment status -i 44d89f46-1a29-4044-ad89-5151213dfcbc

44d89f46-1a29-4044-ad89-5151213dfcbc: FAILED
Created on: 6/27/23 11:05 AM
Detailed Status: <Detailed deployment status>
Deployment Error Stack: List of error codes
Deployment Error Types: List of error types
Failure Cause: Cause
```

log

Usa il `logs` comando per analizzare i log di Greengrass sul tuo dispositivo principale.

Sottocomandi

- get
- elenca-parole chiave
- list-log-files

get

Raccogli, filtra e visualizza i file di registro Greengrass. Questo comando supporta solo file di registro in formato JSON. È possibile specificare il formato di registrazione nella configurazione del nucleo.

Riepilogo

```
greengrass-cli logs get
    [--log-dir path/to/a/log/folder]
    [--log-file path/to/a/log/file]
    [--follow true | false ]
    [--filter <filter> ]
    [--time-window <start-time>,<end-time> ]
```

```
    [--verbose ]
    [--no-color ]
    [--before <value> ]
    [--after <value> ]
    [--syslog ]
    [--max-long-queue-size <value> ]
```

Arguments (Argomenti)

- `--log-dir`,. `-ld` Il percorso della directory in cui verificare la presenza di file di registro, ad esempio***/greengrass/v2*/logs**. Non utilizzare con`--syslog`. Utilizzate un argomento separato per ogni directory aggiuntiva da specificare. È necessario utilizzare almeno uno degli `--log-dir` o`--log-file`. È inoltre possibile utilizzare entrambi gli argomenti in un unico comando.
- `--log-file`,`-lf`. I percorsi delle directory di registro che desideri utilizzare. Utilizzate un argomento separato per ogni directory aggiuntiva da specificare. È necessario utilizzare almeno uno degli `--log-dir` o`--log-file`. È inoltre possibile utilizzare entrambi gli argomenti in un unico comando.
- `--follow`,`-fol`. Mostra gli aggiornamenti del registro man mano che si verificano. La CLI di Greengrass continua a funzionare e legge i log specificati. Se si specifica una finestra temporale, la CLI di Greengrass interrompe il monitoraggio dei log al termine di tutte le finestre temporali.
- `--filter`,. `-f` La parola chiave, le espressioni regolari o la coppia chiave-valore da usare come filtro. Fornite questo valore come stringa, espressione regolare o coppia chiave-valore. Utilizzate un argomento separato per ogni filtro aggiuntivo da specificare.

  Quando vengono valutati, più filtri specificati in un singolo argomento vengono separati da operatori OR e i filtri specificati negli argomenti aggiuntivi vengono combinati con gli operatori AND. Ad esempio, se il comando include`--filter "installed" --filter "name=alpha,name=beta"`, Greengrass CLI filtrerà e visualizzerà i messaggi di registro che contengono sia la parola chiave che una `name` chiave con `installed` i valori o. `alpha` `beta`
- `--time-window`,. `-t` La finestra temporale per la quale mostrare le informazioni del registro. È possibile utilizzare sia i timestamp esatti che gli offset relativi. È necessario fornire queste informazioni nel formato. *`<begin-time>`*`,`*`<end-time>`* Se non si specifica né l'ora di inizio né l'ora di fine, il valore predefinito di tale opzione è la data e l'ora correnti del sistema. Utilizzate un argomento separato per ogni finestra temporale aggiuntiva da specificare.

  La CLI di Greengrass supporta i seguenti formati per i timestamp:

- `yyyy-MM-DD`, ad esempio,. `2020-06-30` L'ora predefinita è 00:00:00 quando si utilizza questo formato.

  `yyyyMMDD`, ad esempio,. `20200630` L'ora predefinita è 00:00:00 quando si utilizza questo formato.

  `HH:mm:ss`, ad esempio,. `15:30:45` La data predefinita è la data di sistema corrente quando si utilizza questo formato.

  `HH:mm:ssSSS`, ad esempio,. `15:30:45` La data predefinita è la data corrente del sistema quando si utilizza questo formato.

  `YYYY-MM-DD'T'HH:mm:ss'Z'`, ad esempio,. `2020-06-30T15:30:45Z`

  `YYYY-MM-DD'T'HH:mm:ss`, ad esempio,`2020-06-30T15:30:45`.

  `yyyy-MM-dd'T'HH:mm:ss.SSS`, ad esempio,`2020-06-30T15:30:45.250`.

  Gli offset relativi specificano uno scostamento del periodo di tempo dall'ora corrente del sistema. La CLI di Greengrass supporta il seguente formato per gli offset relativi:. `+|-[<value>h|hr|hours][value m|min|minutes][value]s|sec|seconds`

  Ad esempio, il seguente argomento per specificare una finestra temporale compresa tra 1 ora e 2 ore 15 minuti prima dell'ora corrente è. `--time-window -2h15min,-1hr`

- `--verbose`. Mostra tutti i campi dei messaggi di registro. Non utilizzare con`--syslog`.
- `--no-color`,`-nc`. Rimuovi la codifica a colori. La codifica a colori predefinita per i messaggi di registro utilizza il testo in grassetto rosso. Supporta solo terminali simili a Unix perché utilizza sequenze di escape ANSI.
- `--before`,`-b`. Il numero di righe da mostrare prima di una voce di registro corrispondente. Il valore predefinito è 0.
- `--after`,. `-a` Il numero di righe da mostrare dopo una voce di registro corrispondente. Il valore predefinito è 0.
- `--syslog`. Elabora tutti i file di registro utilizzando il protocollo syslog definito da RFC3164. Non utilizzare con and. `--log-dir` `--verbose` Il protocollo syslog utilizza il seguente formato:. `"<$Priority>$Timestamp $Host $Logger ($Class): $Message"` Se non si specifica un file di registro, la CLI di Greengrass legge i messaggi di registro dalle seguenti posizioni:`/var/log/messages`, o il. `/var/log/syslog` `/var/log/system.log`

AWS IoT Greengrassattualmente non supporta questa funzionalità sui dispositivi Windows core.

- `--max-log-queue-size`,`-m`. Il numero massimo di voci di registro da allocare alla memoria. Utilizzate questa opzione per ottimizzare l'utilizzo della memoria. L'impostazione predefinita è 100.

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando.

```
$ sudo greengrass-cli logs get --verbose \
    --log-file /greengrass/v2/logs/greengrass.log \
    --filter deployment,serviceName=DeploymentService \
    --filter level=INFO \
    --time-window 2020-12-08T01:11:17,2020-12-08T01:11:22

2020-12-08T01:11:17.615Z [INFO] (pool-2-thread-14)
 com.aws.greengrass.deployment.DeploymentService: Current deployment finished.
 {DeploymentId=44d89f46-1a29-4044-ad89-5151213dfcbc, serviceName=DeploymentService,
 currentState=RUNNING}
2020-12-08T01:11:17.675Z [INFO] (pool-2-thread-14)
 com.aws.greengrass.deployment.IotJobsHelper: Updating status of persisted
 deployment. {Status=SUCCEEDED, StatusDetails={detailed-deployment-
status=SUCCESSFUL}, ThingName=MyThing, JobId=22d89f46-1a29-4044-ad89-5151213dfcbc
```

elenca-parole chiave

Mostra le parole chiave suggerite che è possibile utilizzare per filtrare i file di registro.

Riepilogo

```
greengrass-cli logs list-keywords [arguments]
```

Arguments (Argomenti)

Nessuno

Output

Gli esempi seguenti mostrano l'output prodotto quando si esegue questo comando.

```
$ sudo greengrass-cli logs list-keywords
```

```
Here is a list of suggested keywords for Greengrass log:
level=$str
thread=$str
loggerName=$str
eventType=$str
serviceName=$str
error=$str
```

```
$ sudo greengrass-cli logs list-keywords --syslog

Here is a list of suggested keywords for syslog:
priority=$int
host=$str
logger=$str
class=$str
```

list-log-files

Mostra i file di registro che si trovano in una directory specificata.

Riepilogo

```
greengrass-cli logs list-log-files [arguments]
```

Arguments (Argomenti)

`--log-dir`,`-ld`. Il percorso della directory in cui verificare la presenza di file di registro.

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando.

```
$ sudo greengrass-cli logs list-log-files -ld /greengrass/v2/logs/

/greengrass/v2/logs/aws.greengrass.Nucleus.log
/greengrass/v2/logs/main.log
/greengrass/v2/logs/greengrass.log
Total 3 files found.
```

get-debug-password

Utilizzo dell'`get-debug-password`comando per stampare una password generata casualmente per[componente della console di debug locale](`aws.greengrass.LocalDebugConsole`). La password scade 8 ore dopo la sua generazione.

Riepilogo

```
greengrass-cli get-debug-password
```

Arguments (Argomenti)

Nessuno

Output

L'esempio seguente mostra l'output prodotto quando si esegue questo comando.

```
$ sudo greengrass-cli get-debug-password

Username: debug
Password: bEDp3MOHdj8ou2w5de_sCBI2XAaguy3a8XxREXAMPLE
Password expires at: 2021-04-01T17:01:43.921999931-07:00
The local debug console is configured to use TLS security. The certificate is self-
signed so you will need to bypass your web browser's security warnings to open the
 console.
Before you bypass the security warning, verify that the certificate fingerprint
 matches the following fingerprints.
SHA-256: 15 0B 2C E2 54 8B 22 DE 08 46 54 8A B1 2B 25 DE FB 02 7D 01 4E 4A 56 67 96
 DA A6 CC B1 D2 C4 1B
SHA-1: BC 3E 16 04 D3 80 70 DA E0 47 25 F9 90 FA D6 02 80 3E B5 C1
```

Usa AWS IoT Greengrass Testing Framework

Greengrass Testing Framework (GTF) è una raccolta di elementi costitutivi che supporta end-to-end l'automazione dal punto di vista del cliente. GTF utilizza Cucumber come driver di funzionalità. AWS IoT Greengrass utilizza gli stessi elementi costitutivi per qualificare le modifiche al software su vari dispositivi. Per ulteriori informazioni, consulta Greengrass Testing Framework su Github.

GTF è implementato utilizzando Cucumber, uno strumento utilizzato per eseguire test automatici, per incoraggiare uno sviluppo basato sul comportamento (BDD) dei componenti. In Cucumber, le

caratteristiche di questo sistema sono descritte in un tipo speciale di file chiamato. `feature` Ogni funzionalità è descritta in un formato leggibile dall'uomo chiamato scenari, che sono specifiche che possono essere convertite in test automatici. Ogni scenario è delineato come una serie di passaggi che definiscono le interazioni e i risultati del sistema in esame utilizzando un linguaggio specifico del dominio chiamato Gherkin. Una fase Gherkin è collegata al codice di programmazione utilizzando un metodo chiamato step definition che collega le specifiche al flusso di test. Le definizioni dei passaggi in GTF sono implementate con Java.

Argomenti

Come funziona

AWS IoT Greengrass distribuisce GTF come JAR autonomo composto da diversi moduli Java. Per utilizzare GTF per end-to-end testare i componenti, è necessario implementare i test all'interno di un progetto Java. L'aggiunta del JAR standable di test come dipendenza nel progetto Java consente di utilizzare le funzionalità esistenti del GTF ed estenderle scrivendo casi di test personalizzati. Per eseguire i test case personalizzati, puoi creare il tuo progetto Java ed eseguire il JAR di destinazione con le opzioni di configurazione descritte in. Opzioni di configurazione di Greengrass Testing Framework

JAR standalone GTF

Greengrass utilizza Cloudfront come repository Maven per ospitare diverse versioni del JAR standalone GTF. Per un elenco completo delle versioni GTF, consulta le versioni GTF.

Il JAR standalone GTF include i seguenti moduli. Non è limitato solo a questi moduli. Puoi scegliere ciascuna di queste dipendenze separatamente nel tuo progetto o includerle tutte contemporaneamente nel file JAR standalone di test.

- `aws-greengrass-testing-resources`: Questo modulo fornisce l'astrazione per la gestione del ciclo di vita di una AWS risorsa nel corso di un test. Puoi usarlo per definire le tue AWS risorse

personalizzate usando l'`ResourceSpec`astrazione in modo che GTF possa occuparsi della creazione e della rimozione di tali risorse per te.

- `aws-greengrass-testing-platform`: Questo modulo fornisce l'astrazione a livello di piattaforma per il dispositivo sottoposto a test durante il ciclo di vita del test. Contiene le API utilizzate per interagire con il sistema operativo indipendentemente dalla piattaforma e può essere utilizzato per simulare i comandi in esecuzione nella shell del dispositivo.
- `aws-greengrass-testing-components`: Questo modulo è costituito da componenti di esempio utilizzati per testare le funzionalità principali di Greengrass come implementazioni, IPC e altre funzionalità.
- `aws-greengrass-testing-features`: Questo modulo è composto da passaggi comuni riutilizzabili e relative definizioni che vengono utilizzati per i test nell'ambiente Greengrass.

Argomenti

Changelog

La tabella seguente descrive le modifiche in ogni versione del GTF. Per ulteriori informazioni, consulta la pagina GTF Releases su. GitHub

Versione	Modifiche
1.2.0	Nuove funzionalità • Aggiunge passaggi relativi alla rete per configurare MQTT e la connettività di rete Internet durante i test. • Aggiunge passaggi metrici di sistema per monitorare l'utilizzo della RAM e della CPU del dispositivo.

Versione	Modifiche
	Correzioni di bug e miglioramenti • La fase di distribuzione locale della CLI di Greengrass riprova finché non ha esito positivo. • I test bloccano con garbo il nucleo di Greengrass invece di ucciderlo. • Aggiunge un miglioramento in quanto GTF esegue il polling dell'endpoint Credentials fino a quando le AWS IoT credenziali non sono recuperabili per l'alias thing e role. • Corregge gli artefatti mancanti e le directory di ricette. Questa versione corregge anche le versioni dei componenti mancanti. • Risolve un problema in cui GTF fallisce durante la pulizia dell'immagine docker se l'immagine docker non esiste. • Aggiunge la parola chiave CURRENT come versione del componente.
1.1.0	Nuove funzionalità • Aggiunge la possibilità di installare un componente personalizzato con configurazione. Ciò richiede una ricetta per il componente personali zzato. • Aggiunge la possibilità di aggiornare una distribuzione locale con una configurazione personalizzata. Correzioni di bug e miglioramenti • Risolve il problema di incoerenza della versione GTF del contesto di registro.
1.0.0	Versione iniziale.

Opzioni di configurazione di Greengrass Testing Framework

opzioni di configurazione GTF

Greengrass Testing Framework (GTF) consente di configurare determinati parametri durante il lancio del end-to-end processo di test per orchestrare il flusso di test. È possibile specificare queste opzioni di configurazione come argomenti CLI per il JAR standalone GTF.

La versione 1.1.0 e successive di GTF fornisce le seguenti opzioni di configurazione.

- `additional-plugins`— (Opzionale) Plugin aggiuntivi per Cucumber
- `aws-region`— Si rivolge a endpoint regionali specifici perAWSservizi. L'impostazione predefinita èAWSSDK rileva.
- `credentials-path`— FacoltativoAWSpercorso delle credenziali del profilo. L'impostazione predefinita sono le credenziali rilevate nell'ambiente host.
- `credentials-path-rotation`— Durata di rotazione opzionale perAWScredenziali. Il valore predefinito è 15 minuti o`PT15M`.
- `csr-path`— Il percorso della CSR con cui verrà generato il certificato del dispositivo.
- `device-mode`— Il dispositivo bersaglio sottoposto a test. L'impostazione predefinita è il dispositivo locale.
- `env-stage`— Si rivolge all'ambiente di implementazione di Greengrass. L'impostazione predefinita è la produzione.
- `existing-device-cert-arn`— L'arn di un certificato esistente che desideri utilizzare come certificato di dispositivo per Greengrass.
- `feature-path`— File o directory contenente file di funzionalità aggiuntivi. L'impostazione predefinita è che non vengono utilizzati file di feature aggiuntivi.
- `gg-cli-version`— Sostituisce la versione dell'interfaccia a riga di comando di Greengrass. Il valore predefinito è il valore trovato in`ggc.version`.
- `gg-component-bucket`— Il nome di un bucket Amazon S3 esistente che ospita i componenti Greengrass.
- `gg-component-overrides`— Un elenco di componenti di Greengrass sostituiti.
- `gg-persist`— Un elenco di elementi di test che devono persistere dopo l'esecuzione di un test. Il comportamento predefinito consiste nel non rendere persistente nulla. I valori accettati sono:`aws.resources`,`installed.software`, e`generated.files`.
- `gg-runtime`— Un elenco di valori per influenzare il modo in cui il test interagisce con le risorse di test. Questi valori sostituiscono ig`gg.persist`parametro. Se l'impostazione predefinita è vuota, presuppone che tutte le risorse di test siano gestite per test case, incluso il runtime Greengrass installato. I valori accettati sono:`aws.resources`,`installed.software`, e`generated.files`.
- `ggc-archive`— Il percorso del componente del nucleo di Greengrass archiviato.
- `ggc-install-root`— Directory per installare il componente Greengrass nucleus. I valori predefiniti sono test.temp.path e test run.
- `ggc-log-level`— Imposta il livello di registro del nucleo di Greengrass per l'esecuzione del test. L'impostazione predefinita è «INFO».

- `ggc-tes-rolename`— Il ruolo IAM cheAWS IoT GreengrassCore si assumerà l'accessoAWSservizi. Se non esiste un ruolo con un determinato nome, ne verrà creato uno e una politica di accesso predefinita.
- `ggc-trusted-plugins`— L'elenco separato da virgole dei percorsi (sull'host) dei plugin affidabili che devono essere aggiunti a Greengrass. Per fornire il percorso sul DUT stesso, inserisci il prefisso 'dut: '
- `ggc-user-name`— Il valore PosixUser di user:group per il nucleo Greengrass. Il valore predefinito è il nome utente corrente a cui è stato effettuato l'accesso.
- `ggc-version`— Sostituisce la versione del componente Greengrass nucleus in esecuzione. Il valore predefinito è il valore trovato in ggc.archive.
- `log-level`— Livello di registro dell'esecuzione del test. Il valore predefinito è «INFO».
- `parallel-config`— Set di indice di batch e numero di batch come stringa JSON. Il valore predefinito dell'indice batch è 0 e il numero di batch è 1.
- `proxy-url`— Configura tutti i test per indirizzare il traffico attraverso questo URL.
- `tags`— Esegui solo tag di funzionalità. Può essere intersecato con '&'
- `test-id-prefix`— Un prefisso comune applicato a tutte le risorse specifiche del test, tra cuiAWSnomi e tag delle risorse. L'impostazione predefinita è il prefisso «gg».
- `test-log-path`— Directory che conterrà i risultati dell'intera esecuzione del test. Il valore predefinito è «testResults».
- `test-results-json`— Contrassegno per determinare se viene generato un report Cucumber JSON risultante scritto su disco. Il valore predefinito è true.
- `test-results-log`— Contrassegno per determinare se l'output della console viene generato scritto su disco. Il valore predefinito è false (falso).
- `test-results-xml`— Contrassegno per determinare se un report XML JUnit risultante viene generato e scritto su disco. Il valore predefinito è true.
- `test-temp-path`— Directory per generare artefatti di test locali. Il valore predefinito è una directory temporanea casuale con il prefisso gg-testing.
- `timeout-multiplier`— Moltiplicatore fornito a tutti i timeout dei test. Il valore predefinito è 1.0.

Tutorial: Esegui end-to-end i test utilizzando Greengrass Testing Framework e Greengrass Development Kit

AWS IoT GreengrassTesting Framework (GTF) e Greengrass Development Kit (GDK) offrono agli sviluppatori modi per eseguire i test. end-to-end Puoi completare questo tutorial per inizializzare un progetto GDK con un componente, inizializzare un progetto GDK con un modulo di test e creare un end-to-end test case personalizzato. Dopo aver creato il tuo test case personalizzato, puoi eseguire il test.

In questo tutorial, esegui quanto indicato di seguito:

1. Inizializza un progetto GDK con un componente.
2. Inizializza un progetto GDK con un modulo di test. end-to-end
3. Crea un test case personalizzato.
4. Aggiungi un tag al nuovo test case.
5. Crea il JAR di test.
6. Eseguire il test .

Argomenti

Prerequisiti

Per completare questo tutorial, è necessario quanto segue:

- GDK versione 1.3.0 o successiva

- Java
- Maven
- Git

Fase 1: Inizializza un progetto GDK con un componente

- Inizializza una cartella vuota con un progetto GDK. Scarica il `HelloWorld` componente implementato in Python eseguendo il seguente comando.

  ```
  gdk component init -t HelloWorld -l python -n HelloWorld
  ```

  Questo comando crea una nuova directory denominata `HelloWorld` nella directory corrente.

Fase 2: Inizializza un progetto GDK con un end-to-end modulo di test

- GDK consente di scaricare il modello del modulo di test costituito da un'implementazione di funzionalità e fasi. Esegui il comando seguente per aprire la `HelloWorld` directory e inizializzare il progetto GDK esistente utilizzando un modulo di test.

  ```
  cd HelloWorld
  gdk test-e2e init
  ```

  Questo comando crea una nuova directory denominata `gg-e2e-tests` all'interno della `HelloWorld` directory. Questa directory di test è un progetto Maven che dipende dal JAR standalone di test Greengrass.

Passaggio 3: crea un test case personalizzato

La scrittura di un test case personalizzato consiste sostanzialmente in due passaggi: creare un file di funzionalità con uno scenario di test e implementare le definizioni dei passaggi. Per un esempio di creazione di un test case personalizzato, vediEsempio: crea un test case personalizzato. Utilizza i seguenti passaggi per creare il tuo test case personalizzato:

1. Crea un file di funzionalità con uno scenario di test

   Una funzionalità descrive in genere una funzionalità specifica del software che viene testato. In Cucumber, ogni funzionalità è specificata come un singolo file di funzionalità con un titolo,

una descrizione dettagliata e uno o più esempi di casi specifici chiamati scenari. Ogni scenario è composto da un titolo, una descrizione dettagliata e una serie di passaggi che definiscono le interazioni e i risultati attesi. Gli scenari sono scritti in un formato strutturato utilizzando le parole chiave «given», «when» e «then».

2. Implementa le definizioni dei

   Una definizione di fase collega la fase Gherkin in un linguaggio semplice al codice programmatico. Quando Cucumber identifica un passaggio Gherkin in uno scenario, cercherà una definizione di fase corrispondente da eseguire.

Passaggio 4: aggiungere un tag al nuovo test case

- È possibile assegnare tag alle funzionalità e agli scenari per organizzare il processo di test. È possibile utilizzare i tag per classificare i sottoinsiemi di scenari e anche selezionare gli hook in modo condizionale da eseguire. Le funzionalità e gli scenari possono avere più tag separati da uno spazio.

  In questo esempio, stiamo usando il `HelloWorld` componente.

  Nel file delle funzionalità, aggiungete un nuovo tag denominato `@HelloWorld` accanto al `@Sample` tag.

  ```
  @Sample @HelloWorld
  Scenario: As a developer, I can create a component and deploy it on my device
  ....
  ```

Fase 5: Creare il JAR di test

1. Costruisci il componente. È necessario creare il componente prima di creare il modulo di test.

   ```
   gdk component build
   ```

2. Costruite il modulo di test utilizzando il seguente comando. Questo comando creerà il JAR di test nella `greengrass-build` cartella.

   ```
   gdk test-e2e build
   ```

Passaggio 6: eseguire il test

Quando esegui un test case personalizzato, GTF automatizza il ciclo di vita del test oltre alla gestione delle risorse create durante il test. Innanzitutto esegue il provisioning di un dispositivo in fase di test (DUT) come dispositivo e installa su di esso il software di base Greengrass. AWS IoT Creerà quindi un nuovo componente denominato `HelloWorld` utilizzando la ricetta specificata in quel percorso. Il `HelloWorld` componente viene quindi distribuito sul dispositivo principale tramite una distribuzione di oggetti Greengrass. Verrà quindi verificato se l'implementazione è riuscita. Lo stato di distribuzione verrà modificato `COMPLETED` entro 3 minuti se la distribuzione ha esito positivo.

1. Vai al `gdk-config.json` file nella directory del progetto per indirizzare i test con il `HelloWorld` tag. Aggiorna la `test-e2e` chiave usando il seguente comando.

   ```
   "test-e2e":{
     "gtf_options" : {
          "tags":"HelloWorld"
      }
   }
   ```

2. Prima di eseguire i test, è necessario fornire AWS le credenziali al dispositivo host. GTF utilizza queste credenziali per gestire le AWS risorse durante il processo di test. Assicurati che il ruolo che fornisci disponga delle autorizzazioni necessarie per automatizzare le operazioni necessarie incluse nel test.

   Esegui i seguenti comandi per fornire le credenziali. AWS

   - Linux or Unix

     ```
     export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
     export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
     ```

     Windows Command Prompt (CMD)

     ```
     set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
     set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
     ```

     PowerShell

     ```
     $env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
     ```

```
$env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
```

3. Eseguite il test utilizzando il seguente comando.

   ```
   gdk test-e2e run
   ```

   Questo comando scarica l'ultima versione del nucleo Greengrass nella `greengrass-build` cartella ed esegue i test utilizzandola. Questo comando si rivolge inoltre solo agli scenari con il `HelloWorld` tag e genera un report per tali scenari. Vedrai che le AWS risorse create durante questo test vengono scartate alla fine del test.

Esempio: crea un test case personalizzato

Example

Il modulo di test scaricato nel progetto GDK è costituito da una funzionalità di esempio e da un file di implementazione delle fasi.

Nell'esempio seguente, creiamo un file di funzionalità per testare la funzionalità di distribuzione degli oggetti del software Greengrass. Testiamo parzialmente la funzionalità di questa funzionalità con uno scenario che esegue l'implementazione di un componente tramite GreengrassCloud AWS. Questa è una serie di passaggi che ci aiutano a comprendere le interazioni e i risultati attesi di questo caso d'uso.

1. Create un file di feature

   Passate alla `gg-e2e-tests/src/main/resources/greengrass/features` cartella nella directory corrente. È possibile trovare l'esempio `component.feature` simile all'esempio seguente.

   In questo file di funzionalità, puoi testare la funzionalità di distribuzione degli oggetti del software Greengrass. È possibile testare parzialmente la funzionalità di questa funzionalità con uno scenario che esegue una distribuzione di un componente tramite il cloud Greengrass. Lo scenario è una serie di passaggi che aiutano a comprendere le interazioni e i risultati attesi di questo caso d'uso.

   ```
   Feature: Testing features of Greengrassv2 component

   Background:
       Given my device is registered as a Thing
   ```

```
    And my device is running Greengrass

@Sample
Scenario: As a developer, I can create a component and deploy it on my device
    When I create a Greengrass deployment with components
        HelloWorld | /path/to/recipe/file
    And I deploy the Greengrass deployment configuration
    Then the Greengrass deployment is COMPLETED on the device after 180 seconds
    And I call my custom step
```

GTF contiene le definizioni dei passaggi di tutti i passaggi seguenti, ad eccezione del passaggio denominato:. `And I call my custom step`

2. Implementa le definizioni dei

   Il JAR standalone GTF contiene le definizioni di tutti i passaggi tranne uno:. `And I call my custom step` Puoi implementare questo passaggio nel modulo di test.

   Vai al codice sorgente del file di test. È possibile collegare il passaggio personalizzato utilizzando una definizione di passaggio utilizzando il comando seguente.

```
@And("I call my custom step")
public void customStep() {
    System.out.println("My custom step was called ");
}
```

Tutorial: usa un test di confidenza dalla suite di test di fiducia

AWS IoT GreengrassTesting Framework (GTF) e Greengrass Development Kit (GDK) offrono agli sviluppatori modi per eseguire i test. end-to-end Puoi completare questo tutorial per inizializzare un progetto GDK con un componente, inizializzare un progetto GDK con un modulo di test e utilizzare un end-to-end test di confidenza della suite di test di confidenza. Dopo aver creato il tuo test case personalizzato, puoi eseguire il test.

Un test di confidenza è un test generico fornito da Greengrass che convalida i comportamenti fondamentali dei componenti. Questi test possono essere modificati o estesi per soddisfare esigenze più specifiche dei componenti.

Per questo tutorial useremo un HelloWorld componente. Se stai usando un altro componente, sostituisci il HelloWorld componente con il tuo componente.

In questo tutorial, esegui quanto indicato di seguito:

1. Inizializza un progetto GDK con un componente.
2. Inizializza un progetto GDK con un modulo di test. end-to-end
3. Usa un test della suite di test di fiducia.
4. Aggiungi un tag al nuovo test case.
5. Crea il JAR di test.
6. Eseguire il test .

Argomenti

Prerequisiti

Per completare questo tutorial, è necessario quanto segue:

- GDK versione 1.6.0 o successiva
- Java
- Maven
- Git

Fase 1: Inizializza un progetto GDK con un componente

- Inizializza una cartella vuota con un progetto GDK. Scarica il `HelloWorld` componente implementato in Python eseguendo il seguente comando.

```
gdk component init -t HelloWorld -l python -n HelloWorld
```

Questo comando crea una nuova directory denominata `HelloWorld` nella directory corrente.

Fase 2: Inizializza un progetto GDK con un end-to-end modulo di test

- GDK consente di scaricare il modello del modulo di test costituito da un'implementazione di funzionalità e fasi. Esegui il comando seguente per aprire la `HelloWorld` directory e inizializzare il progetto GDK esistente utilizzando un modulo di test.

  ```
  cd HelloWorld
  gdk test-e2e init
  ```

  Questo comando crea una nuova directory denominata `gg-e2e-tests` all'interno della `HelloWorld` directory. Questa directory di test è un progetto Maven che dipende dal JAR standalone di test Greengrass.

Passaggio 3: utilizzare un test della suite di test di fiducia

La scrittura di un test case di fiducia consiste nell'utilizzare il file di funzionalità fornito e, se necessario, nella modifica degli scenari. Per un esempio di utilizzo di un test di confidenza, vedereEsempio: crea un test case personalizzato. Per utilizzare un test di confidenza, attenersi alla seguente procedura:

- Utilizzate il file di funzionalità fornito.

  Passate alla `gg-e2e-tests/src/main/resources/greengrass/features` cartella nella directory corrente. Apri il `confidenceTest.feature` file di esempio per utilizzare il test di fiducia.

Fase 4: Aggiungere un tag al nuovo test case

- È possibile assegnare tag alle funzionalità e agli scenari per organizzare il processo di test. È possibile utilizzare i tag per classificare i sottoinsiemi di scenari e anche selezionare gli hook in modo condizionale da eseguire. Le funzionalità e gli scenari possono avere più tag separati da uno spazio.

In questo esempio, stiamo usando il `HelloWorld` componente.

Ogni scenario è contrassegnato con`@ConfidenceTest`. Modifica o aggiungi tag se desideri eseguire solo un sottoinsieme della suite di test. Ogni scenario di test è descritto all'inizio di ogni test di confidenza. Lo scenario è una serie di passaggi che aiutano a comprendere le interazioni e i risultati attesi di ogni caso di test. È possibile estendere questi test aggiungendo passaggi personalizzati o modificando quelli esistenti.

```
@ConfidenceTest
Scenario: As a Developer, I can deploy GDK_COMPONENT_NAME to my device and see it
 is working as expected
....
```

Passaggio 5: crea il JAR di test

1. Costruisci il componente. È necessario creare il componente prima di creare il modulo di test.

   ```
   gdk component build
   ```

2. Costruite il modulo di test utilizzando il seguente comando. Questo comando creerà il JAR di test nella `greengrass-build` cartella.

   ```
   gdk test-e2e build
   ```

Passaggio 6: eseguire il test

Quando si esegue un test di confidenza, il GTF automatizza il ciclo di vita del test oltre alla gestione delle risorse create durante il test. Innanzitutto esegue il provisioning di un dispositivo in fase di test (DUT) come dispositivo e installa su di esso il software di base Greengrass. AWS IoT Creerà quindi un nuovo componente denominato `HelloWorld` utilizzando la ricetta specificata in quel percorso. Il `HelloWorld` componente viene quindi distribuito sul dispositivo principale tramite una distribuzione di oggetti Greengrass. Verrà quindi verificato se l'implementazione è riuscita. Lo stato di distribuzione verrà modificato `COMPLETED` entro 3 minuti se la distribuzione ha esito positivo.

1. Vai al `gdk-config.json` file nella directory del progetto per indirizzare i test con il `ConfidenceTest` tag o il tag yo8u specificato nel passaggio 4. Aggiorna la `test-e2e` chiave usando il seguente comando.

```
  "test-e2e":{
    "gtf_options" : {
         "tags":"ConfidenceTest"
     }
  }
```

2. Prima di eseguire i test, è necessario fornire AWS le credenziali al dispositivo host. GTF utilizza queste credenziali per gestire le AWS risorse durante il processo di test. Assicurati che il ruolo che fornisci disponga delle autorizzazioni necessarie per automatizzare le operazioni necessarie incluse nel test.

   Esegui i seguenti comandi per fornire le credenziali. AWS

   - Linux or Unix

     ```
     export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
     export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
     ```

     Windows Command Prompt (CMD)

     ```
     set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
     set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
     ```

     PowerShell

     ```
     $env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
     $env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
     ```

3. Eseguite il test utilizzando il seguente comando.

   ```
   gdk test-e2e run
   ```

   Questo comando scarica l'ultima versione del nucleo Greengrass nella `greengrass-build` cartella ed esegue i test utilizzandola. Questo comando si rivolge inoltre solo agli scenari con il `ConfidenceTest` tag e genera un report per tali scenari. Vedrai che le AWS risorse create durante questo test vengono scartate alla fine del test.

Esempio: utilizza un test di fiducia

Example

Il modulo di test scaricato nel progetto GDK è costituito da un file di funzionalità fornito.

Nell'esempio seguente, utilizziamo un file di funzionalità per testare la funzionalità di distribuzione degli oggetti del software Greengrass. Testiamo parzialmente la funzionalità di questa funzionalità con uno scenario che esegue l'implementazione di un componente tramite GreengrassCloud AWS. Questa è una serie di passaggi che ci aiutano a comprendere le interazioni e i risultati attesi di questo caso d'uso.

- Utilizza il file di funzionalità fornito.

  Passate alla `gg-e2e-tests/src/main/resources/greengrass/features` cartella nella directory corrente. È possibile trovare l'esempio `confidenceTest.feature` simile all'esempio seguente.

```
Feature: Confidence Test Suite

Background:
    Given my device is registered as a Thing
    And my device is running Greengrass

@ConfidenceTest
Scenario: As a Developer, I can deploy GDK_COMPONENT_NAME to my device and see it
 is working as expected
    When I create a Greengrass deployment with components
      | GDK_COMPONENT_NAME | GDK_COMPONENT_RECIPE_FILE |
      | aws.greengrass.Cli | LATEST                    |
    And I deploy the Greengrass deployment configuration
    Then the Greengrass deployment is COMPLETED on the device after 180 seconds
    # Update component state accordingly. Possible states: {RUNNING, FINISHED,
 BROKEN, STOPPING}
    And I verify the GDK_COMPONENT_NAME component is RUNNING using the greengrass-
cli
```

  Ogni scenario di test è descritto all'inizio di ogni test di confidenza. Lo scenario è una serie di passaggi che aiutano a comprendere le interazioni e i risultati attesi di ogni caso di test. È possibile estendere questi test aggiungendo passaggi personalizzati o modificando quelli

esistenti. Ciascuno degli scenari include commenti che consentono di apportare queste modifiche.

Sviluppa AWS IoT Greengrass componenti

Puoi sviluppare e testare componenti sul tuo dispositivo principale Greengrass. Di conseguenza, è possibile creare e iterare il AWS IoT Greengrass software senza interagire con. Cloud AWS Quando completi una versione del componente, puoi caricarla nel cloud, AWS IoT Greengrass in modo che tu e il tuo team possiate distribuire il componente su altri dispositivi del parco macchine. Per ulteriori informazioni su come distribuire i componenti, consulta. Implementazione AWS IoT Greengrass dei componenti sui dispositivi

Ogni componente è composto da una ricetta e da artefatti.

- Ricette

  Ogni componente contiene un file di ricette, che ne definisce i metadati. La ricetta specifica anche i parametri di configurazione del componente, le dipendenze dei componenti, il ciclo di vita e la compatibilità della piattaforma. Il ciclo di vita del componente definisce i comandi che installano, eseguono e spengono il componente. Per ulteriori informazioni, consulta AWS IoT Greengrassriferimento alla ricetta del componente.

  È possibile definire ricette in formato JSON o YAML.

- Artefatti

  I componenti possono avere un numero qualsiasi di artefatti, che sono componenti binari. Gli artefatti possono includere script, codice compilato, risorse statiche e qualsiasi altro file utilizzato da un componente. I componenti possono anche consumare artefatti derivanti dalle dipendenze dei componenti.

AWS IoT Greengrassfornisce componenti predefiniti che è possibile utilizzare nelle applicazioni e distribuire sui dispositivi. Ad esempio, puoi utilizzare il componente stream manager per caricare dati su vari AWS servizi oppure puoi utilizzare il componente CloudWatch metrics per pubblicare metriche personalizzate su Amazon. CloudWatch Per ulteriori informazioni, consulta AWS-componenti forniti.

AWS IoT Greengrasscura un indice dei componenti di Greengrass, chiamato Greengrass Software Catalog. Questo catalogo tiene traccia dei componenti Greengrass sviluppati dalla comunità

Greengrass. Da questo catalogo, puoi scaricare, modificare e distribuire componenti per creare le tue applicazioni Greengrass. Per ulteriori informazioni, consulta Componenti comunitari.

Il software AWS IoT Greengrass Core esegue i componenti come utente e gruppo del sistema, come `ggc_user` `eggc_group`, configurati sul dispositivo principale. Ciò significa che i componenti dispongono delle autorizzazioni di quell'utente del sistema. Se utilizzi un utente di sistema senza una home directory, i componenti non possono utilizzare i comandi di esecuzione o il codice che utilizza una home directory. Ciò significa che non è possibile utilizzare il `pip install some-library --user` comando per installare pacchetti Python, ad esempio. Se hai seguito il tutorial introduttivo per configurare il tuo dispositivo principale, allora l'utente del sistema non dispone di una home directory. Per ulteriori informazioni su come configurare l'utente e il gruppo che eseguono i componenti, consultaConfigurare l'utente che esegue i componenti.

> **Note**
>
> AWS IoT Greengrassutilizza versioni semantiche per i componenti. Le versioni semantiche seguono una delle principali. minore. sistema di numerazione delle patch. Ad esempio, la versione `1.0.0` rappresenta la prima release principale di un componente. Per ulteriori informazioni, consultate la specifica della versione semantica.

Argomenti

Ciclo di vita dei componenti

Il ciclo di vita dei componenti definisce le fasi utilizzate dal software AWS IoT Greengrass Core per installare ed eseguire i componenti. Ogni fase definisce uno script e altre informazioni che specificano

il comportamento del componente. Ad esempio, quando si installa un componente, il software AWS IoT Greengrass Core esegue `Install` lo script del ciclo di vita per quel componente. I componenti dei dispositivi principali hanno i seguenti stati del ciclo di vita:

- `NEW`— La ricetta e gli artefatti del componente vengono caricati sul dispositivo principale, ma il componente non è installato. Dopo che un componente entra in questo stato, esegue lo script di installazione.
- `INSTALLED`— Il componente è installato sul dispositivo principale. Il componente entra in questo stato dopo aver eseguito lo script di installazione.
- `STARTING`— Il componente si avvia sul dispositivo principale. Il componente entra in questo stato quando esegue lo script di avvio. Se l'avvio ha esito positivo, il componente entra nello `RUNNING` stato.
- `RUNNING`— Il componente è in esecuzione sul dispositivo principale. Il componente entra in questo stato quando esegue lo script di esecuzione o quando ha processi in background attivi dallo script di avvio.
- `FINISHED`— Il componente è stato eseguito correttamente e l'esecuzione è stata completata.
- `STOPPING`— Il componente si sta arrestando. Il componente entra in questo stato quando esegue lo script di spegnimento.
- `ERRORED`— Il componente ha riscontrato un errore. Quando il componente entra in questo stato, esegue lo script di ripristino. Quindi, il componente si riavvia per provare a tornare al normale utilizzo. Se il componente entra nello `ERRORED` stato tre volte senza un'esecuzione riuscita, il componente diventa`BROKEN`.
- `BROKEN`— Il componente ha riscontrato errori più volte e non può essere ripristinato. È necessario distribuire nuovamente il componente per risolverlo.

Tipi di componenti

Il tipo di componente specifica in che modo il software AWS IoT Greengrass Core esegue il componente. I componenti possono avere i seguenti tipi:

- Nucleo (`aws.greengrass.nucleus`)

  Il Greengrass nucleus è il componente che fornisce le funzionalità minime del AWS IoT Greengrass software Core. Per ulteriori informazioni, consulta Nucleo Greengrass.
- Plugin (`aws.greengrass.plugin`)

Il nucleo Greengrass esegue un componente plug-in nella stessa Java Virtual Machine (JVM) del nucleo. Il nucleo si riavvia quando si modifica la versione di un componente plug-in su un dispositivo principale. Per installare ed eseguire i componenti del plug-in, è necessario configurare il nucleo Greengrass per l'esecuzione come servizio di sistema. Per ulteriori informazioni, consulta Configurare il nucleo Greengrass come servizio di sistema.

Diversi componenti forniti da AWS sono componenti plug-in, che consentono loro di interfacciarsi direttamente con il nucleo Greengrass. I componenti del plug-in utilizzano lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

- Generico () `aws.greengrass.generic`

  Il nucleo Greengrass esegue gli script del ciclo di vita di un componente generico, se il componente definisce un ciclo di vita.

  Questo tipo è il tipo predefinito per i componenti personalizzati.

- Lambda () `aws.greengrass.lambda`

  Il nucleo Greengrass esegue un componente della funzione Lambda utilizzando il componente di avvio Lambda.

  Quando si crea un componente da una funzione Lambda, il componente ha questo tipo. Per ulteriori informazioni, consulta Esegui AWS Lambda funzioni.

> **Note**
>
> Non è consigliabile specificare il tipo di componente in una ricetta. AWS IoT Greengrassimposta il tipo automaticamente quando crei un componente.

Crea AWS IoT Greengrass componenti

È possibile sviluppare AWS IoT Greengrass componenti personalizzati su un computer di sviluppo locale o su un dispositivo principale Greengrass. AWS IoT Greengrass fornisce l'interfaccia a riga di comando del AWS IoT Greengrass Development Kit (GDK CLI) per aiutarti a creare, creare e pubblicare componenti da modelli di componenti predefiniti e componenti della community. È inoltre possibile eseguire comandi shell incorporati per creare, creare e pubblicare componenti. Scegliete tra le seguenti opzioni per creare componenti Greengrass personalizzati:

- Usa la CLI del Greengrass Development Kit

  Usa la CLI GDK per sviluppare componenti su un computer di sviluppo locale. La CLI GDK compila e impacchetta il codice sorgente dei componenti in una ricetta e in artefatti che puoi pubblicare come componente privato del servizio. AWS IoT Greengrass Puoi configurare la CLI di GDK per aggiornare automaticamente la versione del componente e gli URI degli artefatti quando pubblichi il componente, in modo da non dover aggiornare la ricetta ogni volta. Per sviluppare un componente utilizzando la CLI GDK, puoi iniziare da un modello o da un componente della community dal Greengrass Software Catalog. Per ulteriori informazioni, consulta AWS IoT GreengrassInterfaccia a riga di comando del Development Kit.
- Esegui i comandi della shell incorporati

  È possibile eseguire comandi shell integrati per sviluppare componenti su un computer di sviluppo locale o su un dispositivo principale Greengrass. Utilizzate i comandi della shell per copiare o creare il codice sorgente dei componenti in artefatti. Ogni volta che create una nuova versione di un componente, dovete creare o aggiornare la ricetta con la nuova versione del componente. Quando pubblicate il componente sul AWS IoT Greengrass servizio, dovete aggiornare l'URI per ogni elemento del componente nella ricetta.

Argomenti

Creare un componente (GDK CLI)

Segui le istruzioni in questa sezione per creare e creare un componente utilizzando la CLI GDK.

Sviluppare un componente Greengrass (GDK CLI)

1. Se non l'hai già fatto, installa la CLI GDK sul tuo computer di sviluppo. Per ulteriori informazioni, consulta Installare o aggiornare l'interfaccia a riga di comando del AWS IoT Greengrass Development Kit.
2. Passa alla cartella in cui desideri creare le cartelle dei componenti.

   Linux or Unix

   ```
   mkdir ~/greengrassv2
   ```

```
cd ~/greengrassv2
```

Windows Command Prompt (CMD)

```
mkdir %USERPROFILE%\greengrassv2
cd %USERPROFILE%\greengrassv2
```

PowerShell

```
mkdir ~/greengrassv2
cd ~/greengrassv2
```

3. Scegliete un modello di componente o un componente della community da scaricare. La CLI di GDK scarica il modello o il componente della community, quindi puoi iniziare da un esempio funzionale. Utilizzate il comando component list per recuperare l'elenco dei modelli disponibili o dei componenti della community.
   - Per elencare i modelli dei componenti, esegui il comando seguente. Ogni riga della risposta include il nome e il linguaggio di programmazione di un modello.

     ```
     gdk component list --template
     ```

   - Per elencare i componenti della community, esegui il comando seguente.

     ```
     gdk component list --repository
     ```

4. Crea e passa a una cartella di componenti in cui la CLI di GDK scarica il modello o il componente della community. Sostituisci *HelloWorld*con il nome del componente o un altro nome che ti aiuti a identificare la cartella del componente.

   Linux or Unix

   ```
   mkdir HelloWorld
   cd HelloWorld
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir HelloWorld
   cd HelloWorld
   ```

PowerShell

```
mkdir HelloWorld
cd HelloWorld
```

5. Scarica il modello o il componente della community nella cartella corrente. Usa il comando component init.

   - Per creare una cartella di componenti da un modello, esegui il comando seguente. Sostituisci *HelloWorld* con il nome del modello e sostituisci *python* con il nome del linguaggio di programmazione.

     ```
     gdk component init --template HelloWorld --language python
     ```

   - Per creare una cartella di componenti da un componente della comunità, esegui il comando seguente. Sostituisci *ComponentName* con il nome del componente della community.

     ```
     gdk component init --repository ComponentName
     ```

   > **Note**
   >
   > Se usi GDK CLI v1.0.0, devi eseguire questo comando in una cartella vuota. La CLI GDK scarica il modello o il componente della community nella cartella corrente.
   > Se utilizzi GDK CLI v1.1.0 o versione successiva, puoi specificare `--name` l'argomento per specificare la cartella in cui la CLI di GDK scarica il modello o il componente della community. Se usi questo argomento, specifica una cartella che non esiste. La CLI GDK crea la cartella per te. Se non specificate questo argomento, la CLI di GDK utilizza la cartella corrente, che deve essere vuota.

6. La CLI GDK legge dal file di configurazione GDK CLI, `gdk-config.json` denominato, per creare e pubblicare componenti. Questo file di configurazione esiste nella radice della cartella del componente. Il passaggio precedente crea questo file automaticamente. In questo passaggio, si aggiorna `gdk-config.json` con le informazioni sul componente. Esegui questa operazione:

   a. Aprire `gdk-config.json` in un editor di testo.

   b. (Facoltativo) Modifica il nome del componente. Il nome del componente è la chiave dell'`component`oggetto.

   c. Cambia l'autore del componente.

d. (Facoltativo) Modifica la versione del componente. Specifica una delle seguenti proprietà:

   - `NEXT_PATCH`— Quando scegliete questa opzione, la CLI di GDK imposta la versione quando pubblicate il componente. La CLI GDK interroga AWS IoT Greengrass il servizio per identificare l'ultima versione pubblicata del componente. Quindi, imposta la versione alla versione della patch successiva a quella versione. Se non hai mai pubblicato il componente prima, la CLI di GDK utilizza la versione. `1.0.0`

     Se scegli questa opzione, non puoi utilizzare la CLI di Greengrass per distribuire e testare localmente il componente sul tuo computer di sviluppo locale che esegue il software Core. AWS IoT Greengrass Per abilitare le distribuzioni locali, devi invece specificare una versione semantica.

   - Una versione semantica, ad esempio. **`1.0.0`** Le versioni semantiche utilizzano un major. minore. sistema di numerazione delle patch. Per ulteriori informazioni, consulta la specifica della versione semantica.

     Se sviluppate componenti su un dispositivo Greengrass core su cui desiderate distribuire e testare il componente, scegliete questa opzione. È necessario creare il componente con una versione specifica per creare distribuzioni locali con la Greengrass CLI.

e. (Facoltativo) Modificate la configurazione di compilazione per il componente. La configurazione di build definisce come la CLI di GDK crea il sorgente del componente in artefatti. Scegliete tra le seguenti opzioni per: `build_system`

   - `zip`— Impacchetta la cartella del componente in un file ZIP da definire come unico elemento del componente. Scegliete questa opzione per i seguenti tipi di componenti:
     - Componenti che utilizzano linguaggi di programmazione interpretati, come Python o. JavaScript
     - Componenti che impacchettano file diversi dal codice, come modelli di apprendimento automatico o altre risorse.

     La CLI GDK comprime la cartella del componente in un file zip con lo stesso nome della cartella del componente. Ad esempio, se il nome della cartella del componente è`HelloWorld`, la CLI GDK crea un file zip denominato. `HelloWorld.zip`

> **Note**
>
> Se si utilizza la versione 1.0.0 della CLI di GDK su un dispositivo Windows, i nomi delle cartelle dei componenti e dei file zip devono contenere solo lettere minuscole.

Quando la CLI di GDK comprime la cartella del componente in un file zip, salta i seguenti file:

- Il file `gdk-config.json`
- Il file della ricetta (o) `recipe.json` `recipe.yaml`
- Crea cartelle, come `greengrass-build`

- `maven`— Esegue il `mvn clean package` comando per creare il codice sorgente del componente in artefatti. Scegli questa opzione per i componenti che utilizzano Maven, come i componenti Java.

  Sui dispositivi Windows, questa funzionalità è disponibile per GDK CLI v1.1.0 e versioni successive.

- `gradle`— Esegue il `gradle build` comando per creare il codice sorgente del componente in artefatti. Scegliete questa opzione per i componenti che utilizzano Gradle. Questa funzionalità è disponibile per GDK CLI v1.1.0 e versioni successive.

  Il sistema di `gradle` build supporta Kotlin DSL come file di build. Questa funzionalità è disponibile per GDK CLI v1.2.0 e versioni successive.

- `gradlew`— Esegue il `gradlew` comando per creare il codice sorgente del componente in artefatti. Scegliete questa opzione per i componenti che utilizzano il Gradle Wrapper.

  Questa funzionalità è disponibile per GDK CLI v1.2.0 e versioni successive.

- `custom`— Esegue un comando personalizzato per creare il codice sorgente del componente in una ricetta e in artefatti. Specificate il comando personalizzato nel `custom_build_command` parametro.

f. Se specificate `custom` for`build_system`, aggiungete il `custom_build_command` all'`build`oggetto. In`custom_build_command`, specificate una singola stringa o un elenco di stringhe, dove ogni stringa è una parola nel comando. Ad esempio, per eseguire un

comando di compilazione personalizzato per un componente C++, è possibile specificare. **`["cmake", "--build", "build", "--config", "Release"]`**

g. Se utilizzi GDK CLI v1.1.0 o versione successiva, puoi specificare `--bucket` l'argomento per specificare il bucket S3 in cui la CLI di GDK carica gli artefatti del componente. *Se non specifichi questo argomento, la CLI di GDK viene caricata nel bucket S3 il cui nome bucket-region-accountId è, dove bucket e region sono i valori specificati e AccountID è il tuo IDgdk-config.json.* Account AWS La CLI GDK crea il bucket se non esiste.

   Modifica la configurazione di pubblicazione per il componente. Esegui questa operazione:

   i. Specificate il nome del bucket S3 da utilizzare per ospitare gli artefatti dei componenti.

   ii. Specificate Regione AWS dove la CLI di GDK pubblica il componente.

Al termine di questo passaggio, il `gdk-config.json` file potrebbe essere simile all'esempio seguente.

```
{
  "component": {
    "com.example.PythonHelloWorld": {
      "author": "Amazon",
      "version": "NEXT_PATCH",
      "build": {
        "build_system" : "zip"
      },
      "publish": {
        "bucket": "greengrass-component-artifacts",
        "region": "us-west-2"
      }
    }
  },
  "gdk_version": "1.0.0"
}
```

7. Aggiornate il file di ricetta del componente, denominato `recipe.yaml` `orecipe.json`. Esegui questa operazione:

   a. Se hai scaricato un modello o un componente della community che utilizza il sistema di `zip` compilazione, verifica che il nome dell'artefatto zip corrisponda al nome della cartella del componente. La CLI GDK comprime la cartella del componente in un file zip con lo

stesso nome della cartella del componente. La ricetta contiene il nome dell'artefatto zip nell'elenco degli artefatti dei componenti e negli script del ciclo di vita che utilizzano i file nell'elemento zip. Aggiorna le `Lifecycle` definizioni `Artifacts` e in modo che il nome del file zip corrisponda al nome della cartella del componente. I seguenti esempi di ricette parziali evidenziano il nome del file zip nelle `Lifecycle` definizioni `Artifacts` and.

JSON

```
{
  ...
  "Manifests": [
    {
      "Platform": {
        "os": "all"
      },
      "Artifacts": [
        {
          "URI": "s3://{COMPONENT_NAME}/{COMPONENT_VERSION}/HelloWorld.zip",
          "Unarchive": "ZIP"
        }
      ],
      "Lifecycle": {
        "run": "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
 {configuration:/Message}"
      }
    }
  ]
}
```

YAML

```
---
...
Manifests:
  - Platform:
      os: all
    Artifacts:
      - URI: "s3://BUCKET_NAME/COMPONENT_NAME/
COMPONENT_VERSION/HelloWorld.zip"
        Unarchive: ZIP
    Lifecycle:
```

```
      run: "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
 {configuration:/Message}"
```

b. (Facoltativo) Aggiornate la descrizione del componente, la configurazione predefinita, gli artefatti, gli script del ciclo di vita e il supporto della piattaforma. Per ulteriori informazioni, consulta AWS IoT Greengrassriferimento alla ricetta del componente.

Al termine di questo passaggio, il file delle ricette potrebbe essere simile ai seguenti esempi.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "{COMPONENT_NAME}",
  "ComponentVersion": "{COMPONENT_VERSION}",
  "ComponentDescription": "This is a simple Hello World component written in
 Python.",
  "ComponentPublisher": "{COMPONENT_AUTHOR}",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "Message": "World"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "all"
      },
      "Artifacts": [
        {
          "URI": "s3://{COMPONENT_NAME}/{COMPONENT_VERSION}/HelloWorld.zip",
          "Unarchive": "ZIP"
        }
      ],
      "Lifecycle": {
        "run": "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
 {configuration:/Message}"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: "2020-01-25"
ComponentName: "{COMPONENT_NAME}"
ComponentVersion: "{COMPONENT_VERSION}"
ComponentDescription: "This is a simple Hello World component written in
 Python."
ComponentPublisher: "{COMPONENT_AUTHOR}"
ComponentConfiguration:
  DefaultConfiguration:
    Message: "World"
Manifests:
  - Platform:
      os: all
    Artifacts:
      - URI: "s3://BUCKET_NAME/COMPONENT_NAME/COMPONENT_VERSION/HelloWorld.zip"
        Unarchive: ZIP
    Lifecycle:
      run: "python3 -u {artifacts:decompressedPath}/HelloWorld/main.py
 {configuration:/Message}"
```

8. Sviluppa e costruisci il componente Greengrass. Il comando component build produce una ricetta e degli artefatti nella `greengrass-build` cartella della cartella del componente. Esegui il comando seguente.

```
gdk component build
```

Quando sei pronto per testare il tuo componente, usa la CLI di GDK per pubblicarlo AWS IoT Greengrass sul servizio. Quindi, puoi distribuire il componente sui dispositivi principali Greengrass. Per ulteriori informazioni, consulta Pubblica componenti da distribuire sui tuoi dispositivi principali.

Crea un componente (comandi shell)

Segui le istruzioni in questa sezione per creare cartelle di ricette e artefatti che contengono codice sorgente e artefatti per più componenti.

Sviluppare un componente Greengrass (comandi di shell)

1. Crea una cartella per i tuoi componenti con sottocartelle per ricette e artefatti. Esegui i seguenti comandi sul tuo dispositivo principale Greengrass per creare queste cartelle e passare alla cartella dei componenti. Sostituisci *~/greengrassv2* o *%USERPROFILE%\ greengrassv2* con il percorso della cartella da utilizzare per lo sviluppo locale.

   Linux or Unix

   ```
   mkdir -p ~/greengrassv2/{recipes,artifacts}
   cd ~/greengrassv2
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir %USERPROFILE%\greengrassv2\\recipes, %USERPROFILE%\greengrassv2\\artifacts
   cd %USERPROFILE%\greengrassv2
   ```

   PowerShell

   ```
   mkdir ~/greengrassv2/recipes, ~/greengrassv2/artifacts
   cd ~/greengrassv2
   ```

2. Usa un editor di testo per creare un file di ricette che definisca i metadati, i parametri, le dipendenze, il ciclo di vita e le funzionalità della piattaforma del componente. Includi la versione del componente nel nome del file di ricetta in modo da poter identificare quale ricetta riflette quale versione del componente. Puoi scegliere il formato YAML o JSON per la tua ricetta.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per creare il file.

   JSON

   ```
   nano recipes/com.example.HelloWorld-1.0.0.json
   ```

   YAML

   ```
   nano recipes/com.example.HelloWorld-1.0.0.yaml
   ```

> **Note**
>
> AWS IoT Greengrass utilizza versioni semantiche per i componenti. Le versioni semantiche seguono una delle principali. minore. sistema di numerazione delle patch. Ad esempio, la versione `1.0.0` rappresenta la prima release principale di un componente. Per ulteriori informazioni, consultate la specifica della versione semantica.

3. Definite la ricetta per il componente. Per ulteriori informazioni, consulta AWS IoT Greengrassriferimento alla ricetta del componente.

   La tua ricetta potrebbe essere simile alla seguente ricetta di esempio di Hello World.

   JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.HelloWorld",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "My first AWS IoT Greengrass component.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "Message": "world"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "run": "python3 -u {artifacts:path}/hello_world.py {configuration:/
Message}"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
```

```
        "run": "py -3 -u {artifacts:path}/hello_world.py {configuration:/
Message}"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.HelloWorld
ComponentVersion: '1.0.0'
ComponentDescription: My first AWS IoT Greengrass component.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    Message: world
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      run: |
        python3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
  - Platform:
      os: windows
    Lifecycle:
      run: |
        py -3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
```

Questa ricetta esegue uno script Hello World Python, che potrebbe essere simile allo script di esempio seguente.

```
import sys

message = "Hello, %s!" % sys.argv[1]

# Print the message to stdout, which Greengrass saves in a log file.
print(message)
```

4. Crea una cartella per lo sviluppo della versione del componente. Ti consigliamo di utilizzare una cartella separata per gli artefatti di ogni versione del componente in modo da poter identificare quali artefatti sono per ogni versione del componente. Esegui il comando seguente.

   Linux or Unix

   ```
   mkdir -p artifacts/com.example.HelloWorld/1.0.0
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir artifacts/com.example.HelloWorld/1.0.0
   ```

   PowerShell

   ```
   mkdir artifacts/com.example.HelloWorld/1.0.0
   ```

   > **Important**
   >
   > È necessario utilizzare il seguente formato per il percorso della cartella degli artefatti. Includete il nome e la versione del componente specificati nella ricetta.
   >
   > ```
   > artifacts/componentName/componentVersion/
   > ```

5. Create gli artefatti per il componente nella cartella creata nel passaggio precedente. Gli artefatti possono includere software, immagini e qualsiasi altro file binario utilizzato dal componente.

   Quando il componente è pronto, testalo.

Testare AWS IoT Greengrass i componenti con distribuzioni locali

Se sviluppi un componente Greengrass su un dispositivo principale, puoi creare una distribuzione locale per installarlo e testarlo. Segui i passaggi di questa sezione per creare una distribuzione locale.

Se sviluppi il componente su un computer diverso, ad esempio un computer di sviluppo locale, non puoi creare una distribuzione locale. Pubblica invece il componente sul AWS IoT Greengrass servizio in modo da poterlo distribuire sui dispositivi core Greengrass per testarlo. Per ulteriori informazioni,

consultare Pubblica componenti da distribuire sui tuoi dispositivi principali e Implementazione AWS IoT Greengrass dei componenti sui dispositivi.

Per testare un componente su un dispositivo centrale Greengrass

1. Il dispositivo principale registra eventi come gli aggiornamenti dei componenti. È possibile visualizzare questo file di registro per individuare e risolvere errori relativi al componente, ad esempio una ricetta non valida. Questo file di registro mostra anche i messaggi che il componente stampa in modalità standard out (stdout). Ti consigliamo di aprire una sessione terminale aggiuntiva sul tuo dispositivo principale per osservare i nuovi messaggi di registro in tempo reale. Apri una nuova sessione di terminale, ad esempio tramite SSH, ed esegui il comando seguente per visualizzare i log. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale.

   Linux or Unix

   ```
   sudo tail -f /greengrass/v2/logs/greengrass.log
   ```

   PowerShell

   ```
   gc C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
   ```

   È inoltre possibile visualizzare il file di registro del componente.

   Linux or Unix

   ```
   sudo tail -f /greengrass/v2/logs/com.example.HelloWorld.log
   ```

   PowerShell

   ```
   gc C:\greengrass\v2\logs\com.example.HelloWorld.log -Tail 10 -Wait
   ```

2. Nella sessione terminale originale, esegui il comando seguente per aggiornare il dispositivo principale con il componente. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale e sostituisci *~/greengrassv2* con il percorso della cartella di sviluppo locale.

Linux or Unix

```
sudo /greengrass/v2/bin/greengrass-cli deployment create \
  --recipeDir ~/greengrassv2/recipes \
  --artifactDir ~/greengrassv2/artifacts \
  --merge "com.example.HelloWorld=1.0.0"
```

Windows Command Prompt (CMD)

```
C:\greengrass\v2\bin\greengrass-cli deployment create ^
  --recipeDir %USERPROFILE%\greengrassv2\recipes ^
  --artifactDir %USERPROFILE%\greengrassv2\artifacts ^
  --merge "com.example.HelloWorld=1.0.0"
```

PowerShell

```
C:\greengrass\v2\bin\greengrass-cli deployment create `
  --recipeDir ~/greengrassv2/recipes `
  --artifactDir ~/greengrassv2/artifacts `
  --merge "com.example.HelloWorld=1.0.0"
```

> **Note**
>
> Puoi anche usare il `greengrass-cli deployment create` comando per impostare il valore dei parametri di configurazione del componente. Per ulteriori informazioni, consulta Crea.

3. Utilizzate il `greengrass-cli deployment status` comando per monitorare lo stato di avanzamento della distribuzione del componente.

   Unix or Linux

   ```
   sudo /greengrass/v2/bin/greengrass-cli deployment status \
     -i deployment-id
   ```

   Windows Command Prompt (CMD)

   ```
   C:\greengrass\v2\bin\greengrass-cli deployment status ^
   ```

```
  -i deployment-id
```

PowerShell

```
C:\greengrass\v2\bin\greengrass-cli deployment status `
  -i deployment-id
```

4. Testa il tuo componente mentre funziona sul dispositivo principale Greengrass. Una volta completata questa versione del componente, puoi caricarla sul AWS IoT Greengrass servizio. Quindi, è possibile distribuire il componente su altri dispositivi di base. Per ulteriori informazioni, consulta Pubblica componenti da distribuire sui tuoi dispositivi principali.

Pubblica componenti da distribuire sui tuoi dispositivi principali

Dopo aver creato o completato una versione di un componente, puoi pubblicarla sul AWS IoT Greengrass servizio. Quindi, puoi distribuirlo sui dispositivi principali di Greengrass.

Se utilizzi la CLI del Greengrass Development Kit (GDK CLI) per sviluppare e creare un componente, puoi utilizzare la CLI GDK per pubblicare il componente su. Cloud AWS Altrimenti, utilizzate i comandi di shell incorporati e pubblicate il componente. AWS CLI

È inoltre possibile AWS CloudFormation utilizzarlo per creare componenti e altre AWS risorse dai modelli. Per ulteriori informazioni, consulta Cos'èAWS CloudFormation? e AWS::GreengrassV2::ComponentVersionnella Guida AWS CloudFormation per l'utente.

Argomenti

- Pubblica un componente (GDK CLI)
- Pubblica un componente (comandi shell)

Pubblica un componente (GDK CLI)

Segui le istruzioni in questa sezione per pubblicare un componente utilizzando la CLI GDK. La CLI GDK carica gli artefatti della build in un bucket S3, aggiorna gli URI degli artefatti nella ricetta e crea il componente dalla ricetta. Si specifica il bucket S3 e la regione da utilizzare nel file di configurazione GDK CLI.

Se utilizzi GDK CLI v1.1.0 o versione successiva, puoi specificare `--bucket` l'argomento per specificare il bucket S3 in cui la CLI di GDK carica gli artefatti del componente. *Se non*

specifichi questo argomento, la CLI di GDK viene caricata nel bucket S3 il cui nome bucket-region-accountId è, dove bucket e region sono i valori specificati e AccountID è il tuo IDgdk-config.json. Account AWS La CLI GDK crea il bucket se non esiste.

> **Important**
>
> Per impostazione predefinita, i ruoli principali dei dispositivi non consentono l'accesso ai bucket S3. Se è la prima volta che utilizzi questo bucket S3, devi aggiungere le autorizzazioni al ruolo per consentire ai dispositivi principali di recuperare gli artefatti dei componenti da questo bucket S3. Per ulteriori informazioni, consulta Consenti l'accesso ai bucket S3 per gli artefatti dei componenti.

Per pubblicare un componente Greengrass (GDK CLI)

1. Apri la cartella del componente in un prompt dei comandi o in un terminale.
2. Se non l'hai già fatto, crea il componente Greengrass. Il comando component build produce una ricetta e degli artefatti nella `greengrass-build` cartella della cartella del componente. Esegui il comando seguente.

   ```
   gdk component build
   ```

3. Pubblica il componente in. Cloud AWS Il comando component publish carica gli artefatti del componente su Amazon S3 e aggiorna la ricetta del componente con l'URI di ogni elemento. Quindi, crea il componente nel servizio. AWS IoT Greengrass

   > **Note**
   >
   > AWS IoT Greengrasscalcola il digest di ogni artefatto quando si crea il componente. Ciò significa che non puoi modificare i file degli artefatti nel tuo bucket S3 dopo aver creato un componente. Se lo fai, le distribuzioni che includono questo componente falliranno, perché il file digest non corrisponde. Se modificate un file di artefatti, dovete creare una nuova versione del componente.

Se specifichi `NEXT_PATCH` la versione del componente nel file di configurazione della CLI di GDK, la CLI di GDK utilizza la versione della patch successiva che non esiste già nel servizio. AWS IoT Greengrass

Esegui il comando seguente.

```
gdk component publish
```

L'output indica la versione del componente creata dalla CLI GDK.

Dopo aver pubblicato il componente, puoi distribuirlo sui dispositivi principali. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.

Pubblica un componente (comandi shell)

Utilizzate la procedura seguente per pubblicare un componente utilizzando i comandi di shell e AWS Command Line Interface (AWS CLI). Quando pubblicate un componente, effettuate le seguenti operazioni:

1. Pubblica gli artefatti dei componenti in un bucket S3.
2. Aggiungi l'URI Amazon S3 di ogni elemento alla ricetta del componente.
3. Crea una versione del componente AWS IoT Greengrass dalla ricetta del componente.

> **Note**
>
> Ogni versione del componente che carichi deve essere unica. Assicurati di caricare la versione corretta del componente, perché non puoi modificarla dopo averla caricata.

Puoi seguire questi passaggi per pubblicare un componente dal tuo computer di sviluppo o dal tuo dispositivo principale Greengrass.

Per pubblicare un componente (comandi shell)

1. Se il componente utilizza una versione esistente nel AWS IoT Greengrass servizio, è necessario modificare la versione del componente. Apri la ricetta in un editor di testo, incrementa la versione e salva il file. Scegliete una nuova versione che rifletta le modifiche apportate al componente.

   > **Note**
   >
   > AWS IoT Greengrassutilizza versioni semantiche per i componenti. Le versioni semantiche seguono una delle principali. minore. sistema di numerazione delle patch. Ad esempio, la versione `1.0.0` rappresenta la prima release principale di un componente. Per ulteriori informazioni, consultate la specifica della versione semantica.

2. Se il componente presenta artefatti, procedi come segue:

   a. Pubblica gli artefatti del componente in un bucket S3 nel tuo. Account AWS

      > **Tip**
      >
      > Ti consigliamo di includere il nome e la versione del componente nel percorso dell'artefatto nel bucket S3. Questo schema di denominazione può aiutarti a mantenere gli artefatti utilizzati dalle versioni precedenti del componente, in modo da poter continuare a supportare le versioni precedenti del componente.

      Esegui il comando seguente per pubblicare un file di artefatti in un bucket S3. *Sostituisci DOC-EXAMPLE-BUCKET con il nome del bucket e sostituisci artifacts/com.example. HelloWorld*/1.0.0/artifact.py con il percorso del file degli artefatti.

      ```
      aws s3 cp artifacts/com.example.HelloWorld/1.0.0/artifact.py s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.HelloWorld/1.0.0/artifact.py
      ```

      > **Important**
      >
      > Per impostazione predefinita, i ruoli principali dei dispositivi non consentono l'accesso ai bucket S3. Se è la prima volta che utilizzi questo bucket S3, devi aggiungere le autorizzazioni al ruolo per consentire ai dispositivi principali

di recuperare gli artefatti dei componenti da questo bucket S3. Per ulteriori informazioni, consulta Consenti l'accesso ai bucket S3 per gli artefatti dei componenti.

b. Aggiungi un elenco denominato `Artifacts` alla ricetta del componente se non è presente. L'`Artifacts`elenco viene visualizzato in ogni manifesto, che definisce i requisiti del componente su ciascuna piattaforma supportata (o i requisiti predefiniti del componente per tutte le piattaforme).

c. Aggiungi ogni elemento all'elenco degli artefatti o aggiorna l'URI degli artefatti esistenti. L'URI Amazon S3 è composto dal nome del bucket e dal percorso dell'oggetto artefatto nel bucket. Gli URI Amazon S3 dei tuoi artefatti dovrebbero essere simili all'esempio seguente.

```
s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.HelloWorld/1.0.0/artifact.py
```

Dopo aver completato questi passaggi, la ricetta dovrebbe avere un `Artifacts` elenco simile al seguente.

JSON

```
{
  ...
  "Manifests": [
    {
      "Lifecycle": {
        ...
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/MyGreengrassComponent/1.0.0/artifact.py",
          "Unarchive": "NONE"
        }
      ]
    }
  ]
}
```

> **Note**
>
> È possibile aggiungere l'`"Unarchive": "ZIP"`opzione per un artefatto ZIP per configurare il software AWS IoT Greengrass Core per decomprimere l'artefatto quando il componente viene distribuito.

YAML

```
...
Manifests:
  - Lifecycle:
      ...
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/MyGreengrassComponent/1.0.0/
artifact.py
        Unarchive: NONE
```

> **Note**
>
> È possibile utilizzare l'`Unarchive: ZIP`opzione per configurare il software AWS IoT Greengrass Core per decomprimere un elemento ZIP quando il componente viene distribuito. Per ulteriori informazioni su come utilizzare gli elementi ZIP in un componente, consultate la variabile di ricetta artifacts:DecompressedPath.

Per ulteriori informazioni sulle ricette, consulta AWS IoT Greengrassriferimento alla ricetta del componente.

3. Utilizzate la AWS IoT Greengrass console per creare un componente dal file di ricetta.

   Eseguite il comando seguente per creare il componente da un file di ricette. Questo comando crea il componente e lo pubblica come AWS IoT Greengrass componente privato nel tuoAccount AWS. Sostituisci *path/to/recipeFile con il percorso del file* delle ricette.

   ```
   aws greengrassv2 create-component-version --inline-recipe fileb://path/to/
   recipeFile
   ```

Copia il `arn` codice dalla risposta per verificare lo stato del componente nel passaggio successivo.

> **Note**
>
> AWS IoT Greengrasscalcola il digest di ogni artefatto quando si crea il componente. Ciò significa che non puoi modificare i file degli artefatti nel tuo bucket S3 dopo aver creato un componente. Se lo fai, le distribuzioni che includono questo componente falliranno, perché il file digest non corrisponde. Se modificate un file di artefatti, dovete creare una nuova versione del componente.

4. Ogni componente del AWS IoT Greengrass servizio ha uno stato. Eseguite il comando seguente per confermare lo stato della versione del componente pubblicata in questa procedura. Sostituisci *`com.example. HelloWorld`*e *`1.0.0`* con la versione del componente da interrogare. Sostituisci `arn` con l'ARN del passaggio precedente.

   ```
   aws greengrassv2 describe-component --arn "arn:aws:greengrass:region:account-id:components:com.example.HelloWorld:versions:1.0.0"
   ```

   L'operazione restituisce una risposta che contiene i metadati del componente. I metadati contengono un `status` oggetto che contiene lo stato del componente e gli eventuali errori, se applicabile.

   Quando lo stato del componente è`DEPLOYABLE`, è possibile distribuire il componente sui dispositivi. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.

Interagisci con AWS i servizi

I dispositivi core Greengrass utilizzano certificati X.509 a cui connettersi AWS IoT Core utilizzando i protocolli di autenticazione reciproca TLS. Questi certificati consentono ai dispositivi di interagire AWS IoT senza AWS credenziali, che in genere comprendono un ID della chiave di accesso e una chiave di accesso segreta. Altri AWS servizi richiedono AWS credenziali anziché certificati X.509 per richiamare le operazioni API sugli endpoint del servizio. AWS IoT Coredispone di un provider di credenziali che consente ai dispositivi di utilizzare il certificato X.509 per autenticare le richieste. AWS Il provider di AWS IoT credenziali autentica i dispositivi utilizzando un certificato X.509 e rilascia AWS credenziali sotto forma di token di sicurezza temporaneo con privilegi limitati. I dispositivi possono

utilizzare questo token per firmare e autenticare qualsiasi richiesta. AWS Ciò elimina la necessità di memorizzare AWS le credenziali sui dispositivi core Greengrass. Per ulteriori informazioni, consulta Autorizzazione delle chiamate dirette ai AWS servizi nella Guida per gli AWS IoT Core sviluppatori.

Per recuperare le credenziali da AWS IoT Greengrass, i dispositivi principali utilizzano un alias di ruolo che punta a un AWS IoT ruolo IAM. Questo ruolo IAM è chiamato ruolo di scambio di token. L'alias del ruolo e il ruolo di scambio di token vengono creati quando si installa il software AWS IoT Greengrass Core. Per specificare l'alias di ruolo utilizzato da un dispositivo principale, configura il `iotRoleAlias` parametro di. Nucleo Greengrass

Il fornitore di AWS IoT credenziali assume il ruolo di scambio di token per conto dell'utente per fornire AWS le credenziali ai dispositivi principali. Puoi associare policy IAM appropriate a questo ruolo per consentire ai tuoi dispositivi principali di accedere alle tue AWS risorse, ad esempio componenti, artefatti nei bucket S3. Per ulteriori informazioni su come configurare il ruolo di scambio di token, consulta. Autorizza i dispositivi principali a interagire conAWSservizi

I dispositivi core Greengrass archiviano AWS le credenziali in memoria e le credenziali scadono dopo un'ora per impostazione predefinita. Se il software AWS IoT Greengrass Core si riavvia, deve recuperare nuovamente le credenziali. È possibile utilizzare l'UpdateRoleAliasoperazione per configurare la durata della validità delle credenziali.

AWS IoT Greengrassfornisce un componente pubblico, il componente del servizio di scambio di token, che è possibile definire come dipendenza nel componente personalizzato dall'interazione con AWS i servizi. Il servizio di scambio di token fornisce al componente una variabile di ambiente che definisce l'URI di un server locale che fornisce AWS le credenziali. `AWS_CONTAINER_CREDENTIALS_FULL_URI` Quando crei un client AWS SDK, il client verifica la presenza di questa variabile di ambiente e si connette al server locale per recuperare AWS le credenziali e le utilizza per firmare le richieste API. Ciò consente di utilizzare AWS SDK e altri strumenti per chiamare i AWS servizi nei componenti. Per ulteriori informazioni, consulta Servizio di scambio di token.

> **Important**
>
> Il supporto per l'acquisizione di AWS credenziali in questo modo è stato aggiunto agli AWS SDK il 13 luglio 2016. Il componente deve utilizzare una versione AWS SDK creata a partire da tale data. Per ulteriori informazioni, consulta Using a support AWS SDK nella Amazon Elastic Container Service Developer Guide.

Per acquisire AWS credenziali nel componente personalizzato, definiscilo `aws.greengrass.TokenExchangeService` come dipendenza nella ricetta del componente. La seguente ricetta di esempio definisce un componente che installa boto3 ed esegue uno script Python che utilizza AWS le credenziali del servizio di scambio di token per elencare i bucket Amazon S3.

> **Note**
>
> Per eseguire questo componente di esempio, il dispositivo deve disporre dell'autorizzazione. `s3:ListAllMyBuckets` Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.ListS3Buckets",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that uses the token exchange service to list
 S3 buckets.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.TokenExchangeService": {
      "VersionRequirement": "^2.0.0",
      "DependencyType": "HARD"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "pip3 install --user boto3",
        "run": "python3 -u {artifacts:path}/list_s3_buckets.py"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
```

```
        "install": "pip3 install --user boto3",
        "run": "py -3 -u {artifacts:path}/list_s3_buckets.py"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.ListS3Buckets
ComponentVersion: '1.0.0'
ComponentDescription: A component that uses the token exchange service to list S3
 buckets.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.TokenExchangeService:
    VersionRequirement: '^2.0.0'
    DependencyType: HARD
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install:
        pip3 install --user boto3
      run: |-
        python3 -u {artifacts:path}/list_s3_buckets.py
  - Platform:
      os: windows
    Lifecycle:
      install:
        pip3 install --user boto3
      run: |-
        py -3 -u {artifacts:path}/list_s3_buckets.py
```

Questo componente di esempio esegue il seguente script Python, `list_s3_buckets.py` che elenca i bucket Amazon S3.

```
import boto3
import os
```

```
try:
    print("Creating boto3 S3 client...")
    s3 = boto3.client('s3')
    print("Successfully created boto3 S3 client")
except Exception as e:
    print("Failed to create boto3 s3 client. Error: " + str(e))
    exit(1)

try:
    print("Listing S3 buckets...")
    response = s3.list_buckets()
    for bucket in response['Buckets']:
        print(f'\t{bucket["Name"]}')
    print("Successfully listed S3 buckets")
except Exception as e:
    print("Failed to list S3 buckets. Error: " + str(e))
    exit(1)
```

Esegui un contenitore Docker

Puoi configurare AWS IoT Greengrass i componenti per eseguire un contenitore Docker da immagini archiviate nelle seguenti posizioni:

- Archivi di immagini pubblici e privati in Amazon Elastic Container Registry (Amazon ECR)
- Archivio pubblico di Docker Hub
- Registro pubblico affidabile di Docker
- Bucket S3

Nel componente personalizzato, includi l'URI dell'immagine Docker come artefatto per recuperare l'immagine ed eseguirla sul dispositivo principale. Per le immagini di Amazon ECR e Docker Hub, puoi utilizzare il componente Docker Application Manager per scaricare le immagini e gestire le credenziali per i repository Amazon ECR privati.

Argomenti

- Usa la comunicazione tra processi nei componenti del contenitore Docker
- Usa AWS le credenziali nei componenti del contenitore Docker (Linux)
- Usa lo stream manager nei componenti del contenitore Docker (Linux)

Requisiti

Per eseguire un contenitore Docker in un componente, è necessario quanto segue:

- Un dispositivo principale Greengrass. Se non lo hai, consultare Tutorial: Nozioni di base su AWS IoT Greengrass V2.
- Docker Engine 1.9.1 o versione successiva installato sul dispositivo principale Greengrass. La versione 20.10 è l'ultima versione verificata per funzionare con il software Core. AWS IoT Greengrass È necessario installare Docker direttamente sul dispositivo principale prima di distribuire componenti che eseguono contenitori Docker.

  > **Tip**
  >
  > Puoi anche configurare il dispositivo principale per installare Docker Engine quando il componente viene installato. Ad esempio, lo script di installazione seguente installa Docker Engine prima di caricare l'immagine Docker. Questo script di installazione funziona su distribuzioni Linux basate su Debian, come Ubuntu. Se si configura il componente per installare Docker Engine con questo comando, potrebbe essere necessario impostarlo `true` nello script del ciclo di vita `RequiresPrivilege` per eseguire l'installazione e i comandi. `docker` Per ulteriori informazioni, consulta AWS IoT Greengrassriferimento alla ricetta del componente.
  >
  > ```
  > apt-get install docker-ce docker-ce-cli containerd.io && docker load -i
  >  {artifacts:path}/hello-world.tar
  > ```

- L'utente di sistema che esegue un componente del contenitore Docker deve disporre delle autorizzazioni di root o amministratore oppure è necessario configurare Docker per eseguirlo come utente non root o non amministratore.
  - Sui dispositivi Linux, puoi aggiungere un utente al gruppo senza il quale chiamare i comandi. `docker` `docker` `sudo`
  - Nei dispositivi Windows, è possibile aggiungere un utente al `docker-users` gruppo per richiamare `docker` comandi senza privilegi di amministratore.

Linux or Unix

Per aggiungere `ggc_user` al `docker` gruppo l'utente non root che usi per eseguire i componenti del contenitore Docker, esegui il comando seguente.

```
sudo usermod -aG docker ggc_user
```

Per ulteriori informazioni, consulta Gestire Docker come utente non root.

Windows Command Prompt (CMD)

Per aggiungere al `docker-users` gruppo`ggc_user`, o l'utente che usi per eseguire i componenti del contenitore Docker, esegui il comando seguente come amministratore.

```
net localgroup docker-users ggc_user /add
```

Windows PowerShell

Per aggiungere al `docker-users` gruppo`ggc_user`, o l'utente che usi per eseguire i componenti del contenitore Docker, esegui il comando seguente come amministratore.

```
Add-LocalGroupMember -Group docker-users -Member ggc_user
```

- File a cui accede il componente del contenitore Docker montato come volume nel contenitore Docker.
- Se configuri il software AWS IoT Greengrass Core per utilizzare un proxy di rete, devi configurare Docker per utilizzare lo stesso server proxy.

Oltre a questi requisiti, è necessario soddisfare anche i seguenti requisiti, se applicabili al proprio ambiente:

- Per utilizzare Docker Compose per creare e avviare i tuoi contenitori Docker, installa Docker Compose sul tuo dispositivo principale Greengrass e carica il file Docker Compose in un bucket S3. È necessario archiviare il file Compose in un bucket S3 nello stesso ambiente del componente. Account AWS Regione AWS Per un esempio che utilizza il `docker-compose up` comando in un componente personalizzato, vedi. Esegui un contenitore Docker da un'immagine pubblica in Amazon ECR o Docker Hub

- Se utilizzi un AWS IoT Greengrass proxy di rete, configura il demone Docker per utilizzare un server proxy.
- Se le tue immagini Docker sono archiviate in Amazon ECR o Docker Hub, includi il componente Docker component manager come dipendenza nel tuo componente contenitore Docker. È necessario avviare il demone Docker sul dispositivo principale prima di distribuire il componente.

  Inoltre, includi gli URI dell'immagine come artefatti dei componenti. Gli URI delle immagini devono essere nel formato illustrato negli esempi seguenti: `docker:registry/image[:tag|@digest]`

  - Immagine Amazon ECR privata: `docker:account-id.dkr.ecr.region.amazonaws.com/repository/image[:tag|@digest]`
  - Immagine pubblica di Amazon ECR: `docker:public.ecr.aws/repository/image[:tag|@digest]`
  - Immagine pubblica di Docker Hub: `docker:name[:tag|@digest]`

  Per ulteriori informazioni sull'esecuzione di contenitori Docker da immagini archiviate in archivi pubblici, consulta. Esegui un contenitore Docker da un'immagine pubblica in Amazon ECR o Docker Hub
- Se le tue immagini Docker sono archiviate in un repository privato Amazon ECR, devi includere il componente del servizio di scambio di token come dipendenza nel componente contenitore Docker. Inoltre, il ruolo del dispositivo Greengrass deve consentire le `ecr:GetDownloadUrlForLayer` azioni`ecr:GetAuthorizationToken`, e`ecr:BatchGetImage`, come mostrato nell'esempio seguente di politica IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "ecr:GetAuthorizationToken",
        "ecr:BatchGetImage",
        "ecr:GetDownloadUrlForLayer"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
```

```
}
```

Per informazioni sull'esecuzione di contenitori Docker da immagini archiviate in un repository privato Amazon ECR, consulta. Esegui un contenitore Docker da un'immagine privata in Amazon ECR

- Per utilizzare le immagini Docker archiviate in un repository privato Amazon ECR, l'archivio privato deve trovarsi nello Regione AWS stesso dispositivo principale.
- Se le immagini Docker o i file Compose sono archiviati in un bucket S3, il ruolo del dispositivo Greengrass deve consentire l'`s3:GetObject`autorizzazione per consentire ai dispositivi principali di scaricare le immagini come artefatti componenti, come mostrato nel seguente esempio di politica IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Per informazioni sull'esecuzione di contenitori Docker da immagini archiviate in Amazon S3, consulta. Esegui un contenitore Docker da un'immagine in Amazon S3

- Per utilizzare la comunicazione tra processi (IPC), AWS le credenziali o lo stream manager nel componente del contenitore Docker, devi specificare opzioni aggiuntive quando esegui il contenitore Docker. Per ulteriori informazioni, consulta gli argomenti seguenti:
  - Usa la comunicazione tra processi nei componenti del contenitore Docker
  - Usa AWS le credenziali nei componenti del contenitore Docker (Linux)
  - Usa lo stream manager nei componenti del contenitore Docker (Linux)

Esegui un contenitore Docker da un'immagine pubblica in Amazon ECR o Docker Hub

Questa sezione descrive come creare un componente personalizzato che utilizza Docker Compose per eseguire un contenitore Docker da immagini Docker archiviate in Amazon ECR e Docker Hub.

Per eseguire un contenitore Docker utilizzando Docker Compose

1. Crea e carica un file Docker Compose in un bucket Amazon S3. Assicurati che il ruolo del dispositivo Greengrass consenta l'`s3:GetObject`autorizzazione per consentire al dispositivo di accedere al file Compose. Il file Compose di esempio mostrato nell'esempio seguente include l'immagine Amazon CloudWatch Agent di Amazon ECR e l'immagine MySQL di Docker Hub.

   ```
   version: "3"
   services:
     cloudwatchagent:
       image: "public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest"
     mysql:
       image: "mysql:8.0"
   ```

2. Crea un componente personalizzato sul tuo dispositivo principale. AWS IoT Greengrass La ricetta di esempio mostrata nell'esempio seguente ha le seguenti proprietà:

   - Il componente Docker Application Manager come dipendenza. Questo componente consente di AWS IoT Greengrass scaricare immagini dai repository pubblici di Amazon ECR e Docker Hub.
   - Un elemento componente che specifica un'immagine Docker in un repository Amazon ECR pubblico.
   - Un elemento componente che specifica un'immagine Docker in un repository pubblico di Docker Hub.
   - Un elemento componente che specifica il file Docker Compose che include i contenitori per le immagini Docker che si desidera eseguire.
   - Uno script di esecuzione del ciclo di vita che utilizza docker-compose up per creare e avviare un contenitore dalle immagini specificate.

   JSON

   ```
   {
     "RecipeFormatVersion": "2020-01-25",
   ```

```
  "ComponentName": "com.example.MyDockerComposeComponent",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that uses Docker Compose to run images
 from public Amazon ECR and Docker Hub.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.DockerApplicationManager": {
      "VersionRequirement": "~2.0.0"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "all"
      },
      "Lifecycle": {
        "run": "docker-compose -f {artifacts:path}/docker-compose.yaml up"
      },
      "Artifacts": [
        {
          "URI": "docker:public.ecr.aws/cloudwatch-agent/cloudwatch-
agent:latest"
        },
        {
          "URI": "docker:mysql:8.0"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/folder/docker-compose.yaml"
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.MyDockerComposeComponent
ComponentVersion: '1.0.0'
ComponentDescription: 'A component that uses Docker Compose to run images from
 public Amazon ECR and Docker Hub.'
ComponentPublisher: Amazon
```

```
ComponentDependencies:
  aws.greengrass.DockerApplicationManager:
    VersionRequirement: ~2.0.0
Manifests:
  - Platform:
      os: all
    Lifecycle:
        run: docker-compose -f {artifacts:path}/docker-compose.yaml up
    Artifacts:
      - URI: "docker:public.ecr.aws/cloudwatch-agent/cloudwatch-agent:latest"
      - URI: "docker:mysql:8.0"
      - URI: "s3://DOC-EXAMPLE-BUCKET/folder/docker-compose.yaml"
```

> **Note**
>
> Per utilizzare la comunicazione tra processi (IPC), AWS le credenziali o lo stream manager nel componente del contenitore Docker, devi specificare opzioni aggiuntive quando esegui il contenitore Docker. Per ulteriori informazioni, consulta gli argomenti seguenti:
>
> - Usa la comunicazione tra processi nei componenti del contenitore Docker
> - Usa AWS le credenziali nei componenti del contenitore Docker (Linux)
> - Usa lo stream manager nei componenti del contenitore Docker (Linux)

3. Testa il componente per verificare che funzioni come previsto.

> **Important**
>
> È necessario installare e avviare il demone Docker prima di distribuire il componente.

Dopo aver distribuito il componente localmente, puoi eseguire il comando docker container ls per verificare che il contenitore funzioni.

```
docker container ls
```

4. Quando il componente è pronto, caricalo su per AWS IoT Greengrass distribuirlo su altri dispositivi principali. Per ulteriori informazioni, consulta Pubblica componenti da distribuire sui tuoi dispositivi principali.

Esegui un contenitore Docker da un'immagine privata in Amazon ECR

Questa sezione descrive come creare un componente personalizzato che esegue un contenitore Docker da un'immagine Docker archiviata in un repository privato in Amazon ECR.

Per eseguire un contenitore Docker

1. Crea un componente personalizzato sul tuo dispositivo AWS IoT Greengrass principale. Usa la seguente ricetta di esempio, che ha le seguenti proprietà:

   - Il componente Docker Application Manager come dipendenza. Questo componente consente di AWS IoT Greengrass gestire le credenziali per scaricare immagini da archivi privati.
   - Il componente del servizio di scambio di token come dipendenza. Questo componente consente di AWS IoT Greengrass recuperare AWS le credenziali per interagire con Amazon ECR.
   - Un elemento componente che specifica un'immagine Docker in un repository Amazon ECR privato.
   - Uno script di esecuzione del ciclo di vita che utilizza docker run per creare e avviare un contenitore dall'immagine.

   JSON

   ```
   {
     "RecipeFormatVersion": "2020-01-25",
     "ComponentName": "com.example.MyPrivateDockerComponent",
     "ComponentVersion": "1.0.0",
     "ComponentDescription": "A component that runs a Docker container from a
    private Amazon ECR image.",
     "ComponentPublisher": "Amazon",
     "ComponentDependencies": {
       "aws.greengrass.DockerApplicationManager": {
         "VersionRequirement": "~2.0.0"
       },
       "aws.greengrass.TokenExchangeService": {
         "VersionRequirement": "~2.0.0"
       }
     },
     "Manifests": [
       {
         "Platform": {
   ```

```
        "os": "all"
      },
      "Lifecycle": {
        "run": "docker run account-
id.dkr.ecr.region.amazonaws.com/repository[:tag|@digest]"
      },
      "Artifacts": [
        {
          "URI": "docker:account-
id.dkr.ecr.region.amazonaws.com/repository[:tag|@digest]"
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.MyPrivateDockerComponent
ComponentVersion: '1.0.0'
ComponentDescription: 'A component that runs a Docker container from a private
 Amazon ECR image.'
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.DockerApplicationManager:
    VersionRequirement: ~2.0.0
  aws.greengrass.TokenExchangeService:
    VersionRequirement: ~2.0.0
Manifests:
  - Platform:
      os: all
    Lifecycle:
        run: docker run account-id.dkr.ecr.region.amazonaws.com/repository[:tag|
@digest]
    Artifacts:
      - URI: "docker:account-id.dkr.ecr.region.amazonaws.com/repository[:tag|
@digest]"
```

> **Note**
>
> Per utilizzare la comunicazione tra processi (IPC), AWS le credenziali o lo stream manager nel componente del contenitore Docker, devi specificare opzioni aggiuntive quando esegui il contenitore Docker. Per ulteriori informazioni, consulta gli argomenti seguenti:
>
> - Usa la comunicazione tra processi nei componenti del contenitore Docker
> - Usa AWS le credenziali nei componenti del contenitore Docker (Linux)
> - Usa lo stream manager nei componenti del contenitore Docker (Linux)

2. Testa il componente per verificare che funzioni come previsto.

   > **Important**
   >
   > È necessario installare e avviare il demone Docker prima di distribuire il componente.

   Dopo aver distribuito il componente localmente, puoi eseguire il comando docker container ls per verificare che il contenitore funzioni.

   ```
   docker container ls
   ```

3. Carica il componente su per AWS IoT Greengrass distribuirlo su altri dispositivi principali. Per ulteriori informazioni, consulta Pubblica componenti da distribuire sui tuoi dispositivi principali.

Esegui un contenitore Docker da un'immagine in Amazon S3

Questa sezione descrive come eseguire un contenitore Docker in un componente da un'immagine Docker archiviata in Amazon S3.

Per eseguire un contenitore Docker in un componente da un'immagine in Amazon S3

1. Esegui il comando docker save per creare un backup di un contenitore Docker. Fornisci questo backup come elemento componente su cui eseguire il contenitore. AWS IoT Greengrass Sostituisci *`hello-world`* con il nome dell'immagine e sostituisci *`hello-world.tar`* con il nome del file di archivio da creare.

```
docker save hello-world > artifacts/com.example.MyDockerComponent/1.0.0/hello-world.tar
```

2. Crea un componente personalizzato sul tuo dispositivo AWS IoT Greengrass principale. Usa la seguente ricetta di esempio, che ha le seguenti proprietà:

   - Uno script di installazione del ciclo di vita che utilizza docker load per caricare un'immagine Docker da un archivio.
   - Uno script di esecuzione del ciclo di vita che utilizza docker run per creare e avviare un contenitore dall'immagine. L'`--rm`opzione pulisce il contenitore quando esce.

   JSON

   ```
   {
     "RecipeFormatVersion": "2020-01-25",
     "ComponentName": "com.example.MyS3DockerComponent",
     "ComponentVersion": "1.0.0",
     "ComponentDescription": "A component that runs a Docker container from an
    image in an S3 bucket.",
     "ComponentPublisher": "Amazon",
     "Manifests": [
       {
         "Platform": {
           "os": "linux"
         },
         "Lifecycle": {
           "install": {
             "Script": "docker load -i {artifacts:path}/hello-world.tar"
           },
           "run": {
             "Script": "docker run --rm hello-world"
           }
         }
       }
     ]
   }
   ```

   YAML

   ```
   ---
   ```

```
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.MyS3DockerComponent
ComponentVersion: '1.0.0'
ComponentDescription: 'A component that runs a Docker container from an image in
 an S3 bucket.'
ComponentPublisher: Amazon
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install:
        Script: docker load -i {artifacts:path}/hello-world.tar
      run:
        Script: docker run --rm hello-world
```

> **Note**
>
> Per utilizzare la comunicazione tra processi (IPC), AWS le credenziali o lo stream manager nel componente del contenitore Docker, devi specificare opzioni aggiuntive quando esegui il contenitore Docker. Per ulteriori informazioni, consulta gli argomenti seguenti:
>
> - Usa la comunicazione tra processi nei componenti del contenitore Docker
> - Usa AWS le credenziali nei componenti del contenitore Docker (Linux)
> - Usa lo stream manager nei componenti del contenitore Docker (Linux)

3. Testa il componente per verificare che funzioni come previsto.

   Dopo aver distribuito il componente localmente, puoi eseguire il comando docker container ls per verificare che il contenitore funzioni.

   ```
   docker container ls
   ```

4. Quando il componente è pronto, carica l'archivio di immagini Docker in un bucket S3 e aggiungi il relativo URI alla ricetta del componente. Quindi, puoi caricare il componente su per AWS IoT Greengrass distribuirlo su altri dispositivi principali. Per ulteriori informazioni, consulta Pubblica componenti da distribuire sui tuoi dispositivi principali.

   Al termine, la ricetta del componente dovrebbe essere simile all'esempio seguente.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.MyS3DockerComponent",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that runs a Docker container from an
 image in an S3 bucket.",
  "ComponentPublisher": "Amazon",
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": {
          "Script": "docker load -i {artifacts:path}/hello-world.tar"
        },
        "run": {
          "Script": "docker run --rm hello-world"
        }
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.MyDockerComponent/1.0.0/hello-world.tar"
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.MyS3DockerComponent
ComponentVersion: '1.0.0'
ComponentDescription: 'A component that runs a Docker container from an image in
 an S3 bucket.'
ComponentPublisher: Amazon
Manifests:
```

```
  - Platform:
      os: linux
    Lifecycle:
      install:
        Script: docker load -i {artifacts:path}/hello-world.tar
      run:
        Script: docker run --rm hello-world
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.MyDockerComponent/1.0.0/hello-world.tar
```

Usa la comunicazione tra processi nei componenti del contenitore Docker

È possibile utilizzare la libreria di comunicazione interprocesso (IPC) di Greengrass SDK per dispositivi AWS IoT per comunicare con il nucleo Greengrass, altri componenti Greengrass e. AWS IoT Core Per ulteriori informazioni, consulta Usa il SDK per dispositivi AWS IoT per comunicare con il nucleo Greengrass, altri componenti e AWS IoT Core.

Per utilizzare IPC in un componente del contenitore Docker, è necessario eseguire il contenitore Docker con i seguenti parametri:

- Montate il socket IPC nel contenitore. Il nucleo Greengrass fornisce il percorso del file socket IPC nella variabile di ambiente. `AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT`
- Imposta le variabili `SVCUID` e di `AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT` ambiente sui valori che il nucleo Greengrass fornisce ai componenti. Il componente utilizza queste variabili di ambiente per autenticare le connessioni al nucleo Greengrass.

Example Ricetta di esempio: pubblica un messaggio MQTT su AWS IoT Core (Python)

La seguente ricetta definisce un componente contenitore Docker di esempio su cui pubblica un messaggio MQTT. AWS IoT Core La ricetta ha le seguenti proprietà:

- Una politica di autorizzazione (`accessControl`) che consente al componente di pubblicare messaggi MQTT su tutti gli AWS IoT Core argomenti. Per ulteriori informazioni, vedere Autorizza i componenti a eseguire operazioni IPC Autorizzazione IPC AWS IoT Core MQTT.
- Un elemento componente che specifica un'immagine Docker come archivio TAR in Amazon S3.
- Uno script di installazione del ciclo di vita che carica l'immagine Docker dall'archivio TAR.

- Uno script di esecuzione del ciclo di vita che esegue un contenitore Docker dall'immagine. Il comando Docker run ha i seguenti argomenti:
  - L'`-v`argomento monta il socket IPC Greengrass nel contenitore.
  - I primi due `-e` argomenti impostano le variabili di ambiente richieste nel contenitore Docker.
  - `-e`Gli argomenti aggiuntivi impostano le variabili di ambiente utilizzate in questo esempio.
  - L'`--rm`argomento pulisce il contenitore quando esce.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.python.docker.PublishToIoTCore",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Uses interprocess communication to publish an MQTT
 message to IoT Core.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "topic": "test/topic/java",
      "message": "Hello, World!",
      "qos": "1",
      "accessControl": {
        "aws.greengrass.ipc.mqttproxy": {
          "com.example.python.docker.PublishToIoTCore:pubsub:1": {
            "policyDescription": "Allows access to publish to IoT Core on all
 topics.",
            "operations": [
              "aws.greengrass#PublishToIoTCore"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "all"
```

```
      },
      "Lifecycle": {
        "install": "docker load -i {artifacts:path}/publish-to-iot-core.tar",
        "run": "docker run -v $AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT:
$AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT -e SVCUID -e
 AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT -e MQTT_TOPIC=
\"{configuration:/topic}\" -e MQTT_MESSAGE=\"{configuration:/message}\" -e MQTT_QOS=
\"{configuration:/qos}\" --rm publish-to-iot-core"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.PublishToIoTCore/1.0.0/publish-to-iot-core.tar"
        }
      ]
    }
  ]
}
```

YAML

```
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.python.docker.PublishToIoTCore
ComponentVersion: 1.0.0
ComponentDescription: Uses interprocess communication to publish an MQTT message to
 IoT Core.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    topic: 'test/topic/java'
    message: 'Hello, World!'
    qos: '1'
    accessControl:
      aws.greengrass.ipc.mqttproxy:
        'com.example.python.docker.PublishToIoTCore:pubsub:1':
          policyDescription: Allows access to publish to IoT Core on all topics.
          operations:
            - 'aws.greengrass#PublishToIoTCore'
          resources:
            - '*'
Manifests:
  - Platform:
      os: all
```

```
    Lifecycle:
      install: 'docker load -i {artifacts:path}/publish-to-iot-core.tar'
      run: |
        docker run \
          -v $AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT:
$AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT \
          -e SVCUID \
          -e AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT \
          -e MQTT_TOPIC="{configuration:/topic}" \
          -e MQTT_MESSAGE="{configuration:/message}" \
          -e MQTT_QOS="{configuration:/qos}" \
          --rm publish-to-iot-core
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.PublishToIoTCore/1.0.0/publish-to-iot-core.tar
```

Usa AWS le credenziali nei componenti del contenitore Docker (Linux)

È possibile utilizzare il componente del servizio di scambio di token per interagire con AWS i servizi nei componenti Greengrass. Questo componente fornisce AWS le credenziali derivanti dal ruolo di scambio di token del dispositivo principale utilizzando un server container locale. Per ulteriori informazioni, consulta Interagisci con AWS i servizi.

> **Note**
>
> L'esempio in questa sezione funziona solo su dispositivi core Linux.

Per utilizzare AWS le credenziali del servizio di scambio di token in un componente del contenitore Docker, è necessario eseguire il contenitore Docker con i seguenti parametri:

- Fornisci l'accesso alla rete host utilizzando l'argomento. `--network=host` Questa opzione consente al contenitore Docker di connettersi al servizio di scambio di token locale per recuperare AWS le credenziali. Questo argomento funziona solo su Docker per Linux.

  > **Warning**
  >
  > Questa opzione consente al contenitore di accedere a tutte le interfacce di rete locale sull'host, quindi è meno sicura rispetto all'esecuzione di contenitori Docker senza questo accesso alla rete host. Considera questo aspetto quando sviluppi ed esegui componenti

del contenitore Docker che utilizzano questa opzione. Per ulteriori informazioni, consulta Network: host nella documentazione Docker.

- Imposta le variabili `AWS_CONTAINER_CREDENTIALS_FULL_URI` e di `AWS_CONTAINER_AUTHORIZATION_TOKEN` ambiente sui valori che il nucleo Greengrass fornisce ai componenti. AWS Gli SDK utilizzano queste variabili di ambiente per recuperare AWS le credenziali.

Example Ricetta di esempio: elenca i bucket S3 in un componente contenitore Docker (Python)

La seguente ricetta definisce un esempio di componente del contenitore Docker che elenca i bucket S3 presenti nel tuo. Account AWS La ricetta ha le seguenti proprietà:

- Il componente del servizio di scambio di token come dipendenza. Questa dipendenza consente al componente di recuperare le AWS credenziali per interagire con altri servizi. AWS
- Un elemento componente che specifica un'immagine Docker come archivio tar in Amazon S3.
- Uno script di installazione del ciclo di vita che carica l'immagine Docker dall'archivio TAR.
- Uno script di esecuzione del ciclo di vita che esegue un contenitore Docker dall'immagine. Il comando Docker run ha i seguenti argomenti:
  - L'`--network=host`argomento fornisce al contenitore l'accesso alla rete host, in modo che il contenitore possa connettersi al servizio di scambio di token.
  - L'`-e`argomento imposta le variabili di ambiente richieste nel contenitore Docker.
  - L'`--rm`argomento pulisce il contenitore quando esce.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.python.docker.ListS3Buckets",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Uses the token exchange service to lists your S3
 buckets.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.TokenExchangeService": {
      "VersionRequirement": "^2.0.0",
      "DependencyType": "HARD"
```

```
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "docker load -i {artifacts:path}/list-s3-buckets.tar",
        "run": "docker run --network=host -e AWS_CONTAINER_AUTHORIZATION_TOKEN -e
 AWS_CONTAINER_CREDENTIALS_FULL_URI --rm list-s3-buckets"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.ListS3Buckets/1.0.0/list-s3-buckets.tar"
        }
      ]
    }
  ]
}
```

YAML

```
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.python.docker.ListS3Buckets
ComponentVersion: 1.0.0
ComponentDescription: Uses the token exchange service to lists your S3 buckets.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.TokenExchangeService:
    VersionRequirement: ^2.0.0
    DependencyType: HARD
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: 'docker load -i {artifacts:path}/list-s3-buckets.tar'
      run: |
        docker run \
          --network=host \
          -e AWS_CONTAINER_AUTHORIZATION_TOKEN \
          -e AWS_CONTAINER_CREDENTIALS_FULL_URI \
```

```
            --rm list-s3-buckets
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.ListS3Buckets/1.0.0/list-s3-buckets.tar
```

Usa lo stream manager nei componenti del contenitore Docker (Linux)

È possibile utilizzare il componente stream manager per gestire i flussi di dati nei componenti Greengrass. Questo componente consente di elaborare flussi di dati e trasferire dati IoT ad alto volume a. Cloud AWS AWS IoT Greengrassfornisce un SDK per la gestione dello stream che puoi utilizzare per interagire con il componente stream manager. Per ulteriori informazioni, consulta Gestisci i flussi di dati sui dispositivi core Greengrass.

> **Note**
>
> L'esempio in questa sezione funziona solo su dispositivi core Linux.

Per utilizzare lo stream manager SDK in un componente del contenitore Docker, devi eseguire il contenitore Docker con i seguenti parametri:

- Fornisci l'accesso alla rete host utilizzando l'argomento. `--network=host` Questa opzione consente al contenitore Docker di interagire con il componente stream manager tramite una connessione TLS locale. Questo argomento funziona solo su Docker per Linux

  > **Warning**
  >
  > Questa opzione consente al contenitore di accedere a tutte le interfacce di rete locale sull'host, quindi è meno sicura rispetto all'esecuzione di contenitori Docker senza questo accesso alla rete host. Considera questo aspetto quando sviluppi ed esegui componenti del contenitore Docker che utilizzano questa opzione. Per ulteriori informazioni, consulta Network: host nella documentazione Docker.

- Se configuri il componente stream manager per richiedere l'autenticazione, che è il comportamento predefinito, imposta la variabile di `AWS_CONTAINER_CREDENTIALS_FULL_URI` ambiente sul valore che il nucleo Greengrass fornisce ai componenti. Per ulteriori informazioni, consulta la configurazione dello stream manager.

- Se configuri il componente stream manager per utilizzare una porta non predefinita, usa la comunicazione tra processi (IPC) per ottenere la porta dalla configurazione del componente stream manager. È necessario eseguire il contenitore Docker con opzioni aggiuntive per utilizzare IPC. Per ulteriori informazioni, consulta gli argomenti seguenti:
  - Connect allo stream manager nel codice dell'applicazione
  - Usa la comunicazione tra processi nei componenti del contenitore Docker

Example Ricetta di esempio: trasmetti un file a un bucket S3 in un componente contenitore Docker (Python)

La seguente ricetta definisce un esempio di componente contenitore Docker che crea un file e lo trasmette a un bucket S3. La ricetta ha le seguenti proprietà:

- Il componente stream manager come dipendenza. Questa dipendenza consente al componente di utilizzare lo stream manager SDK per interagire con il componente stream manager.
- Un elemento componente che specifica un'immagine Docker come archivio TAR in Amazon S3.
- Uno script di installazione del ciclo di vita che carica l'immagine Docker dall'archivio TAR.
- Uno script di esecuzione del ciclo di vita che esegue un contenitore Docker dall'immagine. Il comando Docker run ha i seguenti argomenti:
  - L'`--network=host`argomento fornisce al contenitore l'accesso alla rete host, in modo che il contenitore possa connettersi al componente stream manager.
  - Il primo `-e` argomento imposta la variabile di `AWS_CONTAINER_AUTHORIZATION_TOKEN` ambiente richiesta nel contenitore Docker.
  - `-e`Gli argomenti aggiuntivi impostano le variabili di ambiente utilizzate in questo esempio.
  - L'`-v`argomento monta la cartella di lavoro del componente nel contenitore. Questo esempio crea un file nella cartella di lavoro per caricare quel file su Amazon S3 utilizzando stream manager.
  - L'`--rm`argomento pulisce il contenitore quando esce.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.python.docker.StreamFileToS3",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Creates a text file and uses stream manager to stream the
 file to S3.",
```

```
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.StreamManager": {
      "VersionRequirement": "^2.0.0",
      "DependencyType": "HARD"
    }
  },
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "bucketName": ""
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "docker load -i {artifacts:path}/stream-file-to-s3.tar",
        "run": "docker run --network=host -e AWS_CONTAINER_AUTHORIZATION_TOKEN
 -e BUCKET_NAME=\"{configuration:/bucketName}\" -e WORK_PATH=\"{work:path}\" -v
 {work:path}:{work:path} --rm stream-file-to-s3"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.StreamFileToS3/1.0.0/stream-file-to-s3.tar"
        }
      ]
    }
  ]
}
```

YAML

```
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.python.docker.StreamFileToS3
ComponentVersion: 1.0.0
ComponentDescription: Creates a text file and uses stream manager to stream the file
 to S3.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.StreamManager:
```

```
    VersionRequirement: ^2.0.0
    DependencyType: HARD
ComponentConfiguration:
  DefaultConfiguration:
    bucketName: ''
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: 'docker load -i {artifacts:path}/stream-file-to-s3.tar'
      run: |
        docker run \
          --network=host \
          -e AWS_CONTAINER_AUTHORIZATION_TOKEN \
          -e BUCKET_NAME="{configuration:/bucketName}" \
          -e WORK_PATH="{work:path}" \
          -v {work:path}:{work:path} \
          --rm stream-file-to-s3
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.python.docker.StreamFileToS3/1.0.0/stream-file-to-s3.tar
```

AWS IoT Greengrassriferimento alla ricetta del componente

La ricetta del componente è un file che definisce i dettagli, le dipendenze, gli artefatti e i cicli di vita di un componente. Il ciclo di vita del componente specifica i comandi da eseguire per installare, eseguire e spegnere il componente, ad esempio. Il AWS IoT Greengrass core utilizza i cicli di vita definiti nella ricetta per installare ed eseguire i componenti. Il AWS IoT Greengrass servizio utilizza la ricetta per identificare le dipendenze e gli artefatti da distribuire sui dispositivi principali quando si distribuisce il componente.

Nella ricetta, puoi definire dipendenze e cicli di vita unici per ogni piattaforma supportata da un componente. È possibile utilizzare questa funzionalità per distribuire un componente su dispositivi con più piattaforme che hanno requisiti diversi. Puoi anche utilizzarla per AWS IoT Greengrass impedire l'installazione di un componente su dispositivi che non lo supportano.

Ogni ricetta contiene un elenco di manifesti. Ogni manifesto specifica una serie di requisiti della piattaforma, il ciclo di vita e gli artefatti da utilizzare per i dispositivi principali la cui piattaforma soddisfa tali requisiti. Il dispositivo principale utilizza il primo manifesto con i requisiti di piattaforma

che il dispositivo soddisfa. Specificate un manifesto senza requisiti di piattaforma che corrisponda a qualsiasi dispositivo principale.

Puoi anche specificare un ciclo di vita globale che non si trova in un manifesto. Nel ciclo di vita globale, è possibile utilizzare chiavi di selezione che identificano le sottosezioni del ciclo di vita. Quindi, potete specificare queste chiavi di selezione all'interno di un manifesto per utilizzare quelle sezioni del ciclo di vita globale oltre al ciclo di vita del manifesto. Il dispositivo principale utilizza le chiavi di selezione del manifesto solo se il manifesto non definisce un ciclo di vita. È possibile utilizzare la `all` selezione in un manifesto per abbinare le sezioni del ciclo di vita globale senza chiavi di selezione.

Dopo aver selezionato un manifesto corrispondente al dispositivo principale, il software AWS IoT Greengrass Core effettua le seguenti operazioni per identificare le fasi del ciclo di vita da utilizzare:

- Se il manifesto selezionato definisce un ciclo di vita, il dispositivo principale utilizza quel ciclo di vita.
- Se il manifesto selezionato non definisce un ciclo di vita, il dispositivo principale utilizza il ciclo di vita globale. Il dispositivo principale esegue le seguenti operazioni per identificare quali sezioni del ciclo di vita globale utilizzare:
  - Se il manifesto definisce le chiavi di selezione, il dispositivo principale utilizza le sezioni del ciclo di vita globale che contengono le chiavi di selezione del manifesto.
  - Se il manifesto non definisce le chiavi di selezione, il dispositivo principale utilizza le sezioni del ciclo di vita globale che non dispongono di chiavi di selezione. Questo comportamento è equivalente a un manifesto che definisce la `all` selezione.

> **Important**
>
> Un dispositivo principale deve soddisfare almeno i requisiti di piattaforma di un manifesto per installare il componente. Se nessun manifesto corrisponde al dispositivo principale, il software AWS IoT Greengrass Core non installa il componente e la distribuzione fallisce.

È possibile definire ricette in formato JSON o YAML. La sezione degli esempi di ricette include ricette in ogni formato.

Argomenti

- Convalida delle ricette

- Formato della ricetta
- Variabili di ricetta
- Esempi di ricette

Convalida delle ricette

Greengrass convalida la ricetta di un componente JSON o YAML durante la creazione di una versione del componente. Questa convalida della ricetta verifica la presenza di errori comuni nella composizione dei componenti JSON o YAML al fine di prevenire potenziali problemi di implementazione. La convalida verifica la presenza di errori comuni (ad esempio virgole, parentesi e campi mancanti) e verifica che la ricetta sia ben formata.

Se ricevi un messaggio di errore di convalida della ricetta, controlla la ricetta per eventuali virgole, parentesi o campi mancanti. Verifica che non manchi nessun campo esaminando il formato della ricetta.

Formato della ricetta

Quando si definisce una ricetta per un componente, si specificano le seguenti informazioni nel documento della ricetta. La stessa struttura si applica alle ricette nei formati YAML e JSON.

`RecipeFormatVersion`

La versione modello per la ricetta. Scegliete la seguente opzione:

- `2020-01-25`

`ComponentName`

Il nome del componente definito da questa ricetta. Il nome del componente deve essere univoco Account AWS in ogni regione.

> **Suggerimenti**
>
> - Utilizza il formato di nome di dominio inverso per evitare la collisione dei nomi all'interno della tua azienda. Ad esempio, se la tua azienda possiede `example.com` e lavori a un progetto di energia solare, puoi assegnare un nome al tuo componente Hello World. `com.example.solar.HelloWorld` Questo aiuta a evitare le collisioni tra i nomi dei componenti all'interno dell'azienda.

- Evitate il `aws.greengrass` prefisso nei nomi dei componenti. AWS IoT Greengrassutilizza questo prefisso per i componenti pubblici che fornisce. Se scegli lo stesso nome di un componente pubblico, il componente sostituisce quel componente. Quindi, AWS IoT Greengrass fornisce il componente anziché il componente pubblico quando distribuisce componenti che dipendono da quel componente pubblico. Questa funzionalità consente di ignorare il comportamento dei componenti pubblici, ma può anche interrompere altri componenti se non si intende sovrascrivere un componente pubblico.

`ComponentVersion`

La versione del componente. Il valore massimo per i valori principali, secondari e di patch è 999999.

> **Note**
>
> AWS IoT Greengrassutilizza versioni semantiche per i componenti. Le versioni semantiche seguono una delle principali. minore. sistema di numerazione delle patch. Ad esempio, la versione `1.0.0` rappresenta la prima release principale di un componente. Per ulteriori informazioni, consultate la specifica della versione semantica.

`ComponentDescription`

(Facoltativo) La descrizione del componente.

`ComponentPublisher`

L'editore o l'autore del componente.

`ComponentConfiguration`

(Facoltativo) Un oggetto che definisce la configurazione o i parametri per il componente. Si definisce la configurazione predefinita e quindi, quando si distribuisce il componente, è possibile specificare l'oggetto di configurazione da fornire al componente. La configurazione dei componenti supporta parametri e strutture annidati. Questo oggetto contiene le seguenti informazioni:

`DefaultConfiguration`

Un oggetto che definisce la configurazione predefinita per il componente. Tu definisci la struttura di questo oggetto.

> **Note**
>
> AWS IoT Greengrassutilizza JSON per i valori di configurazione. JSON specifica un tipo di numero ma non distingue tra numeri interi e float. Di conseguenza, i valori di configurazione potrebbero essere convertiti in float in. AWS IoT Greengrass Per garantire che il componente utilizzi il tipo di dati corretto, si consiglia di definire i valori di configurazione numerici come stringhe. Quindi, chiedi al componente di analizzarli come numeri interi o float. Ciò garantisce che i valori di configurazione abbiano lo stesso tipo nella configurazione e sul dispositivo principale.

`ComponentDependencies`

(Facoltativo) Un dizionario di oggetti che definiscono ciascuno una dipendenza dal componente. La chiave per ogni oggetto identifica il nome della dipendenza del componente. AWS IoT Greengrassinstalla le dipendenze dei componenti quando il componente viene installato. AWS IoT Greengrassattende l'inizio delle dipendenze prima di avviare il componente. Ogni oggetto contiene le seguenti informazioni:

`VersionRequirement`

Il vincolo di versione semantica in stile npm che definisce le versioni dei componenti compatibili per questa dipendenza. È possibile specificare una versione o un intervallo di versioni. Per ulteriori informazioni, consulta il calcolatore della versione semantica di npm.

`DependencyType`

(Facoltativo) Il tipo di questa dipendenza. Scegliere tra le seguenti opzioni.

- `SOFT`: il componente non si riavvia se la dipendenza cambia stato.
- `HARD`: il componente viene riavviato se la dipendenza cambia stato.

L'impostazione predefinita è `HARD`.

`ComponentType`

(Facoltativo) Il tipo di componente.

> **Note**
>
> Non è consigliabile specificare il tipo di componente in una ricetta. AWS IoT Greengrassimposta il tipo automaticamente quando crei un componente.

Il tipo può essere uno dei seguenti tipi:

- `aws.greengrass.generic`— Il componente esegue comandi o fornisce artefatti.
- `aws.greengrass.lambda`— Il componente esegue una funzione Lambda utilizzando il componente Lambda launcher. Il `ComponentSource` parametro specifica l'ARN della funzione Lambda eseguita da questo componente.

  Non è consigliabile utilizzare questa opzione, poiché è impostata da AWS IoT Greengrass quando si crea un componente da una funzione Lambda. Per ulteriori informazioni, consulta Esegui AWS Lambda funzioni.
- `aws.greengrass.plugin`— Il componente viene eseguito nella stessa Java Virtual Machine (JVM) del nucleo Greengrass. Se si distribuisce o si riavvia un componente del plug-in, il nucleo Greengrass si riavvia.

  I componenti del plug-in utilizzano lo stesso file di registro del nucleo Greengrass. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

  Non è consigliabile utilizzare questa opzione nelle ricette dei componenti, poiché è destinata ai componenti AWS forniti e scritti in Java che si interfacciano direttamente con il nucleo Greengrass. Per ulteriori informazioni su quali componenti pubblici sono plugin, consulta. AWS-componenti forniti
- `aws.greengrass.nucleus`— Il componente del nucleo. Per ulteriori informazioni, consulta Nucleo Greengrass.

  Non è consigliabile utilizzare questa opzione nelle ricette dei componenti. È destinato al componente Greengrass nucleus, che fornisce la funzionalità minima del AWS IoT Greengrass software Core.

L'impostazione predefinita è `aws.greengrass.generic` quando si crea un componente da una ricetta o `aws.greengrass.lambda` quando si crea un componente da una funzione Lambda.

Per ulteriori informazioni, consulta Tipi di componenti.

`ComponentSource`

(Facoltativo) L'ARN della funzione Lambda eseguita da un componente.

Non è consigliabile specificare l'origine del componente in una ricetta. AWS IoT Greengrassimposta questo parametro per te quando crei un componente da una funzione Lambda. Per ulteriori informazioni, consulta Esegui AWS Lambda funzioni.

`Manifests`

Un elenco di oggetti che definiscono ciascuno il ciclo di vita del componente, i parametri e i requisiti per una piattaforma. Se un dispositivo principale soddisfa i requisiti di piattaforma di più manifesti, AWS IoT Greengrass utilizza il primo manifest a cui il dispositivo principale soddisfa. Per garantire che i dispositivi principali utilizzino il manifesto corretto, definisci innanzitutto i manifesti con requisiti di piattaforma più rigorosi. Un manifesto che si applica a tutte le piattaforme deve essere l'ultimo manifesto dell'elenco.

> **Important**
>
> Un dispositivo principale deve soddisfare i requisiti di piattaforma di almeno un manifesto per installare il componente. Se nessun manifesto corrisponde al dispositivo principale, il software AWS IoT Greengrass Core non installa il componente e la distribuzione fallisce.

Ogni oggetto contiene le seguenti informazioni:

`Name`

(Facoltativo) Un nome descrittivo per la piattaforma definita da questo manifesto.

Se ometti questo parametro, AWS IoT Greengrass crea un nome dalla piattaforma `os` e `architecture`.

`Platform`

(Facoltativo) Un oggetto che definisce la piattaforma a cui si applica questo manifesto. Omettete questo parametro per definire un manifesto applicabile a tutte le piattaforme.

Questo oggetto specifica le coppie chiave-valore relative alla piattaforma su cui viene eseguito un dispositivo principale. Quando si distribuisce questo componente, il software AWS IoT Greengrass Core confronta queste coppie chiave-valore con gli attributi della piattaforma sul dispositivo principale. Il software AWS IoT Greengrass Core definisce sempre `os` e `architecture` potrebbe definire attributi aggiuntivi. È possibile specificare attributi di piattaforma personalizzati per un dispositivo principale quando si distribuisce il componente Greengrass nucleus. Per ulteriori informazioni, consulta il parametro platform overrides del componente Greengrass nucleus.

Per ogni coppia chiave-valore, puoi specificare uno dei seguenti valori:

- Un valore esatto, ad esempio o. `linux` `windows` I valori esatti devono iniziare con una lettera o un numero.
- `*`, che corrisponde a qualsiasi valore. Ciò corrisponde anche quando un valore non è presente.
- Un'espressione regolare in stile Java, ad esempio. `/windows|linux/` L'espressione regolare deve iniziare e terminare con un carattere barra (). `/` Ad esempio, l'espressione regolare `/.+/` corrisponde a qualsiasi valore non vuoto.

Questo oggetto contiene le seguenti informazioni:

`os`

(Facoltativo) Il nome del sistema operativo per la piattaforma supportata da questo manifesto. Le piattaforme comuni includono i seguenti valori:

- `linux`
- `windows`
- `darwin` (macOS)

`architecture`

(Facoltativo) L'architettura del processore per la piattaforma supportata da questo manifesto. Le architetture comuni includono i seguenti valori:

- `amd64`
- `arm`
- `aarch64`
- `x86`

`architecture.detail`

(Facoltativo) I dettagli dell'architettura del processore per la piattaforma supportata da questo manifesto. I dettagli comuni dell'architettura includono i seguenti valori:

- `arm61`
- `arm71`
- `arm81`

`key`

(Facoltativo) Un attributo di piattaforma definito per questo manifesto. Sostituisci la *`chiave`* con il nome dell'attributo della piattaforma. Il software AWS IoT Greengrass Core abbina

questo attributo della piattaforma alle coppie chiave-valore specificate nella configurazione del componente Greengrass nucleus. Per ulteriori informazioni, consulta il parametro platform overrides del componente Greengrass nucleus.

> **Tip**
>
> Utilizzate il formato inverso dei nomi di dominio per evitare la collisione dei nomi all'interno della vostra azienda. Ad esempio, se la tua azienda possiede `example.com` e lavori a un progetto radiofonico, puoi assegnare un nome a un attributo di piattaforma personalizzato. `com.example.radio.RadioModule` Questo aiuta a evitare le collisioni tra i nomi degli attributi della piattaforma all'interno dell'azienda.

Ad esempio, è possibile definire un attributo di piattaforma per specificare un manifesto diverso in base al modulo radio disponibile su un dispositivo principale. `com.example.radio.RadioModule` Ogni manifesto può includere diversi artefatti che si applicano a diverse configurazioni hardware, in modo da distribuire il set minimo di software sul dispositivo principale.

`Lifecycle`

Un oggetto o una stringa che definisce come installare ed eseguire il componente sulla piattaforma definita da questo manifest. È inoltre possibile definire un ciclo di vita globale applicabile a tutte le piattaforme. Il dispositivo principale utilizza il ciclo di vita globale solo se il manifesto da utilizzare non specifica un ciclo di vita.

> **Note**
>
> Questo ciclo di vita viene definito all'interno di un manifesto. Le fasi del ciclo di vita specificate qui si applicano solo alla piattaforma definita da questo manifesto. È inoltre possibile definire un ciclo di vita globale applicabile a tutte le piattaforme.

Questo oggetto o stringa contiene le seguenti informazioni:

`Setenv`

(Facoltativo) Un dizionario di variabili di ambiente da fornire a tutti gli script del ciclo di vita. È possibile sovrascrivere queste variabili di ambiente `Setenv` in ogni script del ciclo di vita.

`install`

(Facoltativo) Un oggetto o una stringa che definisce lo script da eseguire durante l'installazione del componente. Il software AWS IoT Greengrass Core esegue inoltre questa fase del ciclo di vita a ogni avvio del software.

Se lo `install` script termina con un codice di successo, il componente entra nello stato. `INSTALLED`

Questo oggetto o stringa contiene le seguenti informazioni:

`Script`

Lo script da eseguire.

`RequiresPrivilege`

(Facoltativo) È possibile eseguire lo script con i privilegi di root. Se impostate questa opzione su`true`, il software AWS IoT Greengrass Core esegue questo script del ciclo di vita come root anziché come utente di sistema configurato per eseguire questo componente. L'impostazione predefinita è `false`.

`Skipif`

(Facoltativo) Il controllo per determinare se eseguire o meno lo script. È possibile definire se un file eseguibile si trova sul percorso o se esiste un file. Se l'output è vero, il software AWS IoT Greengrass Core salta il passaggio. Scegliete uno dei seguenti controlli:

- `onpath` *`runnable`*— Controlla se un runnable è presente sul percorso di sistema. Ad esempio, usa **`onpath python3`** per saltare questo passaggio del ciclo di vita se Python 3 è disponibile.
- `exists` *`file`*— Controlla se esiste un file. Ad esempio, utilizzare **`exists /tmp/my-configuration.db`** per saltare questa fase del ciclo di vita, se `/tmp/my-configuration.db` presente.

`Timeout`

(Facoltativo) Il periodo di tempo massimo, in secondi, che lo script può essere eseguito prima che il software AWS IoT Greengrass Core termini il processo.

Impostazione predefinita: 120 secondi

`Setenv`

(Facoltativo) Il dizionario delle variabili di ambiente da fornire allo script. Queste variabili di ambiente hanno la precedenza sulle variabili fornite. `Lifecycle.Setenv`

`run`

(Facoltativo) Un oggetto o una stringa che definisce lo script da eseguire all'avvio del componente.

Il componente entra nello `RUNNING` stato quando viene eseguita questa fase del ciclo di vita. Se lo `run` script termina con un codice di successo, il componente entra nello stato. `STOPPING` Se viene specificato uno `shutdown` script, viene eseguito; altrimenti il componente entra nello `FINISHED` stato.

I componenti che dipendono da questo componente vengono avviati durante l'esecuzione di questa fase del ciclo di vita. Per eseguire un processo in background, ad esempio un servizio utilizzato dai componenti dipendenti, utilizzate invece la fase del `startup` ciclo di vita.

Quando distribuisci componenti con un `run` ciclo di vita, il dispositivo principale può segnalare che l'implementazione è completa non appena viene eseguito questo script del ciclo di vita. Di conseguenza, l'implementazione può essere completa e riuscita anche se lo script del `run` ciclo di vita fallisce subito dopo l'esecuzione. Se desideri che lo stato della distribuzione dipenda dal risultato dello script di avvio del componente, utilizza invece la fase del `startup` ciclo di vita.

> **Note**
>
> È possibile definirne solo uno `startup` o `run` il ciclo di vita.

Questo oggetto o stringa contiene le seguenti informazioni:

`Script`

Lo script da eseguire.

`RequiresPrivilege`

(Facoltativo) È possibile eseguire lo script con i privilegi di root. Se impostate questa opzione su`true`, il software AWS IoT Greengrass Core esegue questo script del ciclo

di vita come root anziché come utente di sistema configurato per eseguire questo componente. L'impostazione predefinita è `false`.

`Skipif`

(Facoltativo) Il controllo per determinare se eseguire o meno lo script. È possibile definire se un file eseguibile si trova sul percorso o se esiste un file. Se l'output è vero, il software AWS IoT Greengrass Core salta il passaggio. Scegliete uno dei seguenti controlli:

- `onpath` *`runnable`*— Controlla se un runnable è presente sul percorso di sistema. Ad esempio, usa **`onpath python3`** per saltare questo passaggio del ciclo di vita se Python 3 è disponibile.
- `exists` *`file`*— Controlla se esiste un file. Ad esempio, utilizzare **`exists /tmp/my-configuration.db`** per saltare questa fase del ciclo di vita, se `/tmp/my-configuration.db` presente.

`Timeout`

(Facoltativo) Il periodo di tempo massimo, in secondi, che lo script può essere eseguito prima che il software AWS IoT Greengrass Core termini il processo.

Per impostazione predefinita, questa fase del ciclo di vita non prevede il timeout. Se ometti questo timeout, lo `run` script viene eseguito fino alla sua chiusura.

`Setenv`

(Facoltativo) Il dizionario delle variabili di ambiente da fornire allo script. Queste variabili di ambiente hanno la precedenza sulle variabili fornite. `Lifecycle.Setenv`

`startup`

(Facoltativo) Un oggetto o una stringa che definisce il processo in background da eseguire all'avvio del componente.

Viene utilizzato `startup` per eseguire un comando che deve terminare correttamente o aggiornare lo stato del componente `RUNNING` prima che i componenti dipendenti possano avviarsi. Utilizzate l'operazione UpdateStateIPC per impostare lo stato del componente su `RUNNING` o `ERRORED` quando il componente avvia uno script che non esce. Ad esempio, è possibile definire un `startup` passaggio che avvia il processo MySQL con. `/etc/init.d/mysqld start`

Il componente entra nello `STARTING` stato quando viene eseguita questa fase del ciclo di vita. Se lo `startup` script termina con un codice di successo, il componente entra nello stato. `RUNNING` Quindi, i componenti dipendenti possono avviarsi.

Quando si distribuiscono componenti con un `startup` ciclo di vita, il dispositivo principale può segnalare l'implementazione come completa dopo la chiusura o la segnalazione dello stato di questo script del ciclo di vita. In altre parole, lo stato della distribuzione è valido `IN_PROGRESS` fino alla chiusura o alla segnalazione di uno stato da parte degli script di avvio di tutti i componenti.

> **Note**
>
> È possibile definirne solo uno `startup` o `run` un ciclo di vita.

Questo oggetto o stringa contiene le seguenti informazioni:

`Script`

Lo script da eseguire.

`RequiresPrivilege`

(Facoltativo) È possibile eseguire lo script con i privilegi di root. Se impostate questa opzione su`true`, il software AWS IoT Greengrass Core esegue questo script del ciclo di vita come root anziché come utente di sistema configurato per eseguire questo componente. L'impostazione predefinita è `false`.

`Skipif`

(Facoltativo) Il controllo per determinare se eseguire o meno lo script. È possibile definire se un file eseguibile si trova sul percorso o se esiste un file. Se l'output è vero, il software AWS IoT Greengrass Core salta il passaggio. Scegliete uno dei seguenti controlli:

- `onpath` *`runnable`*— Controlla se un runnable è presente sul percorso di sistema. Ad esempio, usa **`onpath python3`** per saltare questo passaggio del ciclo di vita se Python 3 è disponibile.
- `exists` *`file`*— Controlla se esiste un file. Ad esempio, utilizzare **`exists /tmp/my-configuration.db`** per saltare questa fase del ciclo di vita, se `/tmp/my-configuration.db` presente.

`Timeout`

(Facoltativo) Il periodo di tempo massimo, in secondi, che lo script può essere eseguito prima che il software AWS IoT Greengrass Core termini il processo.

Impostazione predefinita: 120 secondi

`Setenv`

(Facoltativo) Il dizionario delle variabili di ambiente da fornire allo script. Queste variabili di ambiente hanno la precedenza sulle variabili fornite. `Lifecycle.Setenv`

`shutdown`

(Facoltativo) Un oggetto o una stringa che definisce lo script da eseguire quando il componente si spegne. Utilizzate il ciclo di vita di spegnimento per eseguire il codice che desiderate eseguire quando il componente è nello stato in cui si trova. `STOPPING` Il ciclo di vita dello spegnimento può essere utilizzato per interrompere un processo avviato dagli script o. `startup` `run`

Se avviate un processo in background in`startup`, utilizzate il `shutdown` passaggio per interrompere tale processo quando il componente si spegne. Ad esempio, è possibile definire un `shutdown` passaggio che interrompa il processo MySQL con. `/etc/init.d/mysqld stop`

Lo `shutdown` script viene eseguito dopo che il componente è entrato nello `STOPPING` stato. Se lo script viene completato correttamente, il componente entra nello `FINISHED` stato.

Questo oggetto o stringa contiene le seguenti informazioni:

`Script`

Lo script da eseguire.

`RequiresPrivilege`

(Facoltativo) È possibile eseguire lo script con i privilegi di root. Se impostate questa opzione su`true`, il software AWS IoT Greengrass Core esegue questo script del ciclo di vita come root anziché come utente di sistema configurato per eseguire questo componente. L'impostazione predefinita è `false`.

`Skipif`

(Facoltativo) Il controllo per determinare se eseguire o meno lo script. È possibile definire se un file eseguibile si trova sul percorso o se esiste un file. Se l'output è vero, il software AWS IoT Greengrass Core salta il passaggio. Scegliete uno dei seguenti controlli:

- `onpath` *`runnable`*— Controlla se un runnable è presente sul percorso di sistema. Ad esempio, usa **`onpath python3`** per saltare questo passaggio del ciclo di vita se Python 3 è disponibile.
- `exists` *`file`*— Controlla se esiste un file. Ad esempio, utilizzare **`exists /tmp/my-configuration.db`** per saltare questa fase del ciclo di vita, se `/tmp/my-configuration.db` presente.

`Timeout`

(Facoltativo) Il periodo di tempo massimo, in secondi, che lo script può essere eseguito prima che il software AWS IoT Greengrass Core termini il processo.

Impostazione predefinita: 15 secondi.

`Setenv`

(Facoltativo) Il dizionario delle variabili di ambiente da fornire allo script. Queste variabili di ambiente hanno la precedenza sulle variabili fornite. `Lifecycle.Setenv`

`recover`

(Facoltativo) Un oggetto o una stringa che definisce lo script da eseguire quando il componente rileva un errore.

Questo passaggio viene eseguito quando un componente entra nello `ERRORED` stato. Se il componente diventa `ERRORED` tre volte senza riprendersi correttamente, passa allo `BROKEN` stato del componente. Per riparare un `BROKEN` componente, è necessario distribuirlo nuovamente.

Questo oggetto o stringa contiene le seguenti informazioni:

`Script`

Lo script da eseguire.

`RequiresPrivilege`

(Facoltativo) È possibile eseguire lo script con i privilegi di root. Se impostate questa opzione su`true`, il software AWS IoT Greengrass Core esegue questo script del ciclo di vita come root anziché come utente di sistema configurato per eseguire questo componente. L'impostazione predefinita è `false`.

`Skipif`

(Facoltativo) Il controllo per determinare se eseguire o meno lo script. È possibile definire se un file eseguibile si trova sul percorso o se esiste un file. Se l'output è vero, il software AWS IoT Greengrass Core salta il passaggio. Scegliete uno dei seguenti controlli:

- `onpath` *`runnable`*— Controlla se un runnable è presente sul percorso di sistema. Ad esempio, usa **`onpath python3`** per saltare questo passaggio del ciclo di vita se Python 3 è disponibile.
- `exists` *`file`*— Controlla se esiste un file. Ad esempio, utilizzare **`exists /tmp/my-configuration.db`** per saltare questa fase del ciclo di vita, se `/tmp/my-configuration.db` presente.

`Timeout`

(Facoltativo) Il periodo di tempo massimo, in secondi, che lo script può essere eseguito prima che il software AWS IoT Greengrass Core termini il processo.

Default: 60 secondi.

`Setenv`

(Facoltativo) Il dizionario delle variabili di ambiente da fornire allo script. Queste variabili di ambiente hanno la precedenza sulle variabili fornite. `Lifecycle.Setenv`

`bootstrap`

(Facoltativo) Un oggetto o una stringa che definisce uno script che richiede il riavvio del software AWS IoT Greengrass Core o del dispositivo principale. Ciò consente di sviluppare un componente che esegua un riavvio dopo aver installato gli aggiornamenti del sistema operativo o gli aggiornamenti di runtime, ad esempio.

> **Note**
>
> Per installare aggiornamenti o dipendenze che non richiedono il riavvio del software o del dispositivo AWS IoT Greengrass Core, utilizza il ciclo di vita di installazione.

Questa fase del ciclo di vita viene eseguita prima della fase del ciclo di vita di installazione nei seguenti casi, quando il software Core distribuisce il componente: AWS IoT Greengrass

- Il componente viene distribuito sul dispositivo principale per la prima volta.
- La versione del componente cambia.
- Lo script di bootstrap cambia a seguito di un aggiornamento della configurazione del componente.

Dopo che il software AWS IoT Greengrass Core ha completato la fase di bootstrap per tutti i componenti che hanno una fase di bootstrap in una distribuzione, il software si riavvia.

> **Important**
>
> È necessario configurare il software AWS IoT Greengrass Core come servizio di sistema per riavviare il software AWS IoT Greengrass Core o il dispositivo principale. Se non configuri il software AWS IoT Greengrass Core come servizio di sistema, il software non verrà riavviato. Per ulteriori informazioni, consulta Configurare il nucleo Greengrass come servizio di sistema.

Questo oggetto o stringa contiene le seguenti informazioni:

`BootstrapOnRollback`

> **Note**
>
> Quando questa funzionalità è abilitata, `BootstrapOnRollback` verrà eseguita solo per i componenti che hanno completato o tentato di eseguire le fasi del ciclo di vita di bootstrap come parte di una distribuzione di destinazione non riuscita. Questa funzionalità è disponibile per le versioni 2.12.0 e successive di Greengrass nucleus.

(Facoltativo) È possibile eseguire le fasi del ciclo di vita di bootstrap come parte di una distribuzione di rollback. Se imposti questa opzione su`true`, verranno eseguite le fasi del ciclo di vita di bootstrap definite all'interno di una distribuzione di rollback. Quando una distribuzione fallisce, la versione precedente del ciclo di vita di bootstrap del componente verrà eseguita nuovamente durante una distribuzione di rollback.

L'impostazione predefinita è `false`.

`Script`

Lo script da eseguire. Il codice di uscita di questo script definisce l'istruzione di riavvio. Utilizza i seguenti codici di uscita:

- `0`— Non riavviare il software AWS IoT Greengrass Core o il dispositivo principale. Il software AWS IoT Greengrass Core continua a riavviarsi dopo l'avvio di tutti i componenti.
- `100`— Richiesta di riavvio del software AWS IoT Greengrass Core.
- `101`— Richiesta di riavvio del dispositivo principale.

I codici di uscita da 100 a 199 sono riservati a comportamenti speciali. Gli altri codici di uscita rappresentano errori di script.

`RequiresPrivilege`

(Facoltativo) È possibile eseguire lo script con i privilegi di root. Se impostate questa opzione su`true`, il software AWS IoT Greengrass Core esegue questo script del ciclo di vita come root anziché come utente di sistema configurato per eseguire questo componente. L'impostazione predefinita è `false`.

`Timeout`

(Facoltativo) Il periodo di tempo massimo, in secondi, che lo script può essere eseguito prima che il software AWS IoT Greengrass Core termini il processo.

Impostazione predefinita: 120 secondi

`Setenv`

(Facoltativo) Il dizionario delle variabili di ambiente da fornire allo script. Queste variabili di ambiente hanno la precedenza sulle variabili fornite. `Lifecycle.Setenv`

`Selections`

(Facoltativo) Un elenco di chiavi di selezione che specificano le sezioni del ciclo di vita globale da eseguire per questo manifesto. Nel ciclo di vita globale, è possibile definire le fasi del ciclo di vita con chiavi di selezione a qualsiasi livello per selezionare le sottosezioni del ciclo di vita. Quindi, il dispositivo principale utilizza le sezioni che corrispondono ai tasti di selezione in questo manifesto. Per ulteriori informazioni, consulta gli esempi del ciclo di vita globale.

> **Important**
>
> Il dispositivo principale utilizza le selezioni dal ciclo di vita globale solo se questo manifesto non definisce un ciclo di vita.

È possibile specificare la chiave di `all` selezione per eseguire sezioni del ciclo di vita globale che non dispongono di chiavi di selezione.

`Artifacts`

(Facoltativo) Un elenco di oggetti che definiscono ciascuno un artefatto binario per il componente sulla piattaforma definito da questo manifesto. Ad esempio, è possibile definire codice o immagini come artefatti.

Quando il componente viene distribuito, il software AWS IoT Greengrass Core scarica l'elemento in una cartella sul dispositivo principale. È inoltre possibile definire gli artefatti come file di archivio che il software estrae dopo averli scaricati.

È possibile utilizzare le variabili di ricetta per ottenere i percorsi delle cartelle in cui gli artefatti si installano sul dispositivo principale.

- File normali: utilizzate la variabile artifacts:path recipe per ottenere il percorso della cartella che contiene gli artefatti. Ad esempio, specificate `{artifacts:path}/my_script.py` in una ricetta di ottenere il percorso di un artefatto che ha l'URI. `s3://DOC-EXAMPLE-BUCKET/path/to/my_script.py`
- Archivi estratti: utilizzate la variabile di ricetta artifacts:DecompressedPath per ottenere il percorso della cartella che contiene gli elementi dell'archivio estratti. Il software AWS IoT Greengrass Core estrae ogni archivio in una cartella con lo stesso nome dell'archivio. Ad esempio, specificate `{artifacts:decompressedPath}/my_archive/my_script.py` in una ricetta il percorso dell'`my_script.py`elemento dell'archivio contenente l'URI. `s3://DOC-EXAMPLE-BUCKET/path/to/my_archive.zip`

> **Note**
>
> Quando sviluppate un componente con un elemento di archivio su un dispositivo centrale locale, potreste non avere un URI per quell'artefatto. Per testare il componente con un'`Unarchive`opzione che estrae l'artefatto, specificate un URI in cui il nome del file corrisponda al nome del file dell'artefatto di archivio. È possibile specificare l'URI in cui si prevede di caricare l'elemento dell'archivio oppure è possibile specificare un nuovo URI segnaposto. Ad esempio, per estrarre l'`my_archive.zip`artefatto durante una distribuzione locale, è possibile specificare. `s3://DOC-EXAMPLE-BUCKET/my_archive.zip`

Ogni oggetto contiene le seguenti informazioni:

`URI`

L'URI di un artefatto in un bucket S3. Il software AWS IoT Greengrass Core recupera l'artefatto da questo URI quando il componente viene installato, a meno che l'artefatto non esista già sul dispositivo. Ogni elemento deve avere un nome di file univoco all'interno di ogni manifesto.

`Unarchive`

(Facoltativo) Il tipo di archivio da decomprimere. Seleziona una delle opzioni seguenti:

- `NONE`— Il file non è un archivio da decomprimere. Il software AWS IoT Greengrass Core installa l'artefatto in una cartella sul dispositivo principale. Puoi usare la variabile artifacts:path recipe per ottenere il percorso di questa cartella.
- `ZIP`— Il file è un archivio ZIP. Il software AWS IoT Greengrass Core estrae l'archivio in una cartella con lo stesso nome dell'archivio. È possibile utilizzare la variabile di ricetta Artifacts:DecompressedPath per ottenere il percorso della cartella che contiene questa cartella.

L'impostazione predefinita è `NONE`.

`Permission`

(Facoltativo) Un oggetto che definisce le autorizzazioni di accesso da impostare per questo file di artefatti. È possibile impostare il permesso di lettura e il permesso di esecuzione.

> **Note**
>
> Non è possibile impostare l'autorizzazione di scrittura, poiché il software AWS IoT Greengrass Core non consente ai componenti di modificare i file degli artefatti nella cartella artifacts. Per modificare un file di artefatti in un componente, copiatelo in un'altra posizione o pubblicate e distribuite un nuovo file di artefatti.

Se definite un artefatto come un archivio da decomprimere, il software AWS IoT Greengrass Core imposta queste autorizzazioni di accesso sui file che decomprime dall'archivio. Il software AWS IoT Greengrass Core imposta le autorizzazioni di accesso della cartella su for e. `ALL` `Read` `Execute` Ciò consente ai componenti di visualizzare i file decompressi nella cartella. Per impostare le autorizzazioni su singoli file dall'archivio, è possibile impostare le autorizzazioni nello script del ciclo di vita di installazione.

Questo oggetto contiene le seguenti informazioni:

`Read`

(Facoltativo) L'autorizzazione di lettura da impostare per questo file di artefatti. Per consentire ad altri componenti di accedere a questo elemento, ad esempio i componenti che dipendono da questo componente, specificare. `ALL` Seleziona una delle opzioni seguenti:

- `NONE`— Il file non è leggibile.
- `OWNER`— Il file è leggibile dall'utente di sistema configurato per eseguire questo componente.
- `ALL`— Il file è leggibile da tutti gli utenti.

L'impostazione predefinita è `OWNER`.

`Execute`

(Facoltativo) L'autorizzazione di esecuzione da impostare per questo file di artefatti. L'`Execute`autorizzazione implica l'`Read`autorizzazione. Ad esempio, se si specifica `ALL` for`Execute`, tutti gli utenti possono leggere ed eseguire questo file di artefatti.

Seleziona una delle opzioni seguenti:

- `NONE`— Il file non è eseguibile.

- `OWNER`— Il file può essere eseguito dall'utente di sistema configurato per eseguire il componente.
- `ALL`— Il file è eseguibile da tutti gli utenti.

L'impostazione predefinita è `NONE`.

`Digest`

(Sola lettura) L'hash del digest crittografico dell'artefatto. Quando create un componente, AWS IoT Greengrass utilizza un algoritmo di hash per calcolare un hash del file dell'artefatto. Quindi, quando si distribuisce il componente, il nucleo di Greengrass calcola l'hash dell'elemento scaricato e lo confronta con questo digest per verificare l'artefatto prima dell'installazione. Se l'hash non corrisponde al digest, la distribuzione fallisce.

Se impostate questo parametro, AWS IoT Greengrass sostituisce il valore impostato al momento della creazione del componente.

`Algorithm`

(Sola lettura) L'algoritmo hash AWS IoT Greengrass utilizzato per calcolare l'hash digest dell'artefatto.

Se impostate questo parametro, AWS IoT Greengrass sostituisce il valore impostato durante la creazione del componente.

`Lifecycle`

Un oggetto che definisce come installare ed eseguire il componente. Il dispositivo principale utilizza il ciclo di vita globale solo se il manifesto da utilizzare non specifica un ciclo di vita.

> **Note**
>
> Questo ciclo di vita viene definito all'esterno di un manifesto. È inoltre possibile definire un ciclo di vita del manifesto che si applica alle piattaforme che corrispondono a quel manifesto.

Nel ciclo di vita globale, è possibile specificare i cicli di vita che vengono eseguiti per determinate chiavi di selezione specificate in ogni manifesto. Le chiavi di selezione sono stringhe che identificano le sezioni del ciclo di vita globale da eseguire per ogni manifesto.

La chiave `all` di selezione è l'impostazione predefinita per qualsiasi sezione senza chiave di selezione. Ciò significa che è possibile specificare la chiave di `all` selezione in un manifesto per eseguire le sezioni del ciclo di vita globale senza chiavi di selezione. Non è necessario specificare la chiave di `all` selezione nel ciclo di vita globale.

Se un manifesto non definisce un ciclo di vita o chiavi di selezione, per impostazione predefinita il dispositivo principale utilizza la selezione. `all` Ciò significa che in questo caso, il dispositivo principale utilizza le sezioni del ciclo di vita globale che non utilizzano chiavi di selezione.

Questo oggetto contiene le stesse informazioni del ciclo di vita del manifesto, ma è possibile specificare chiavi di selezione a qualsiasi livello per selezionare sottosezioni del ciclo di vita.

> **Tip**
>
> Si consiglia di utilizzare solo lettere minuscole per ogni chiave di selezione per evitare conflitti tra le chiavi di selezione e le chiavi del ciclo di vita. Le chiavi del ciclo di vita iniziano con una lettera maiuscola.

Example Esempio di ciclo di vita globale con tasti di selezione di primo livello

```
Lifecycle:
  key1:
    install:
      Skipif: either onpath executable or exists file
      Script: command1
  key2:
    install:
      Script: command2
  all:
    install:
      Script: command3
```

Example Esempio di ciclo di vita globale con tasti di selezione di livello inferiore

```
Lifecycle:
  install:
    Script:
      key1: command1
      key2: command2
      all: command3
```

Example Esempio di ciclo di vita globale con più livelli di tasti di selezione

```
Lifecycle:
  key1:
    install:
      Skipif: either onpath executable or exists file
      Script: command1
  key2:
    install:
      Script: command2
  all:
    install:
      Script:
        key3: command3
        key4: command4
        all: command5
```

Variabili di ricetta

Le variabili di ricetta espongono le informazioni del componente e del nucleo correnti da utilizzare nelle ricette. Ad esempio, potete utilizzare una variabile recipe per passare i parametri di configurazione dei componenti a un'applicazione eseguita in uno script del ciclo di vita.

È possibile utilizzare le variabili di ricetta nelle seguenti sezioni delle ricette dei componenti:

- Definizioni del ciclo di vita.
- Definizioni di configurazione dei componenti, se si utilizza Greengrass nucleus v2.6.0 o versione successiva e si imposta l'opzione di configurazione su. interpolateComponentConfiguration`true` È inoltre possibile utilizzare le variabili recipes quando si distribuiscono gli aggiornamenti della configurazione dei componenti.

Le variabili Recipe utilizzano la `{recipe_variable}` sintassi. Le parentesi arricciate indicano una variabile di ricetta.

AWS IoT Greengrasssupporta le seguenti variabili di ricetta:

component_dependency_name`:configuration:`*json_pointer*

Il valore di un parametro di configurazione per il componente definito da questa ricetta o per un componente da cui dipende questo componente.

È possibile utilizzare questa variabile per fornire un parametro a uno script eseguito nel ciclo di vita del componente.

> **Note**
>
> AWS IoT Greengrasssupporta questa variabile di ricetta solo nelle definizioni del ciclo di vita dei componenti.

Questa variabile di ricetta ha i seguenti input:

- `component_dependency_name`— (Facoltativo) Il nome della dipendenza del componente da interrogare. Omettete questo segmento per interrogare il componente definito da questa ricetta. È possibile specificare solo dipendenze dirette.
- `json_pointer`— Il puntatore JSON al valore di configurazione da valutare. I puntatori JSON iniziano con una barra. `/` Per identificare un valore in una configurazione di componenti annidati, utilizzate forward slashes (`/`) per separare le chiavi per ogni livello della configurazione. È possibile utilizzare un numero come chiave per specificare un indice in un elenco. Per ulteriori informazioni, consulta la specifica del puntatore JSON.

  AWS IoT GreengrassCore utilizza i puntatori JSON per le ricette in formato YAML.

Il puntatore JSON può fare riferimento ai seguenti tipi di nodi:

- Un nodo di valore. AWS IoT Greengrass Core sostituisce la variabile recipe con la rappresentazione in stringa del valore. I valori nulli vengono convertiti in una `null` stringa.
- Un nodo oggetto. AWS IoT Greengrass Core sostituisce la variabile recipe con la rappresentazione di stringa JSON serializzata di quell'oggetto.
- Nessun nodo. AWS IoT Greengrass Core non sostituisce la variabile recipe.

Ad esempio, la variabile `{configuration:/Message}` recipe recupera il valore della `Message` chiave nella configurazione del componente. La variabile `{com.example.MyComponentDependency:configuration:/server/port}` recipe recupera il valore di `port` nell'oggetto di `server` configurazione di una dipendenza del componente.

`component_dependency_name``:artifacts:path`

Il percorso principale degli artefatti per il componente definito da questa ricetta o per un componente da cui dipende questo componente.

Quando un componente viene installato, AWS IoT Greengrass copia gli artefatti del componente nella cartella esposta da questa variabile. È possibile utilizzare questa variabile per identificare la posizione di uno script da eseguire nel ciclo di vita del componente, ad esempio.

La cartella in questo percorso è di sola lettura. Per modificare i file degli artefatti, copiate i file in un'altra posizione, ad esempio nella directory di lavoro corrente (o). `$PWD .` Quindi, modificate i file lì.

Per leggere o eseguire un artefatto da una dipendenza di un componente, l'artefatto o l'autorizzazione di `Read` tale elemento devono essere. `Execute ALL` Per ulteriori informazioni, consultate le autorizzazioni degli artefatti definite nella ricetta del componente.

Questa variabile di ricetta ha i seguenti input:

- `component_dependency_name`— (Facoltativo) Il nome della dipendenza del componente da interrogare. Omettete questo segmento per interrogare il componente definito da questa ricetta. È possibile specificare solo dipendenze dirette.

`component_dependency_name``:artifacts:decompressedPath`

Il percorso principale degli elementi dell'archivio decompresso per il componente definito da questa ricetta o per un componente da cui dipende questo componente.

Quando un componente viene installato, AWS IoT Greengrass decomprime gli elementi dell'archivio del componente nella cartella esposta da questa variabile. È possibile utilizzare questa variabile per identificare la posizione di uno script da eseguire nel ciclo di vita del componente, ad esempio.

Ogni elemento viene decompresso in una cartella all'interno del percorso decompresso, dove la cartella ha lo stesso nome dell'elemento meno la sua estensione. Ad esempio, un elemento ZIP denominato viene decompresso nella cartella. `models.zip` `{artifacts:decompressedPath}/models`

La cartella in questo percorso è di sola lettura. Per modificare i file degli artefatti, copiate i file in un'altra posizione, ad esempio nella directory di lavoro corrente (o). `$PWD .` Quindi, modificate i file lì.

Per leggere o eseguire un artefatto da una dipendenza di un componente, l'artefatto o l'autorizzazione di `Read` tale elemento devono essere. `Execute ALL` Per ulteriori informazioni, consultate le autorizzazioni degli artefatti definite nella ricetta del componente.

Questa variabile di ricetta ha i seguenti input:

- `component_dependency_name`— (Facoltativo) Il nome della dipendenza del componente da interrogare. Omettete questo segmento per interrogare il componente definito da questa ricetta. È possibile specificare solo dipendenze dirette.

`component_dependency_name``:work:path`

Questa funzionalità è disponibile per la versione 2.0.4 e successive del componente Greengrass nucleus.

Il percorso di lavoro per il componente definito da questa ricetta o per un componente da cui dipende questo componente. Il valore di questa variabile di ricetta è equivalente all'output della variabile di `$PWD` ambiente e del comando pwd quando viene eseguito dal contesto del componente.

È possibile utilizzare questa variabile recipe per condividere file tra un componente e una dipendenza.

La cartella in questo percorso è leggibile e scrivibile dal componente definito da questa ricetta e da altri componenti che funzionano come lo stesso utente e gruppo.

Questa variabile di ricetta ha i seguenti input:

- `component_dependency_name`— (Facoltativo) Il nome della dipendenza del componente da interrogare. Omettete questo segmento per interrogare il componente definito da questa ricetta. È possibile specificare solo dipendenze dirette.

`kernel:rootPath`

Il percorso AWS IoT Greengrass principale principale.

`iot:thingName`

Questa funzionalità è disponibile per la versione 2.3.0 e successive del componente Greengrass nucleus.

Il nome del dispositivo principale. AWS IoT

Esempi di ricette

Puoi fare riferimento ai seguenti esempi di ricette per aiutarti a creare ricette per i tuoi componenti.

AWS IoT Greengrasscura un indice dei componenti di Greengrass, chiamato Greengrass Software Catalog. Questo catalogo tiene traccia dei componenti Greengrass sviluppati dalla comunità

Greengrass. Da questo catalogo, puoi scaricare, modificare e distribuire componenti per creare le tue applicazioni Greengrass. Per ulteriori informazioni, consulta Componenti comunitari.

Argomenti

- Ricetta dei componenti Hello World
- Esempio di componente runtime Python
- Ricetta del componente che specifica diversi campi

Ricetta dei componenti Hello World

La seguente ricetta descrive un componente Hello World che esegue uno script Python. Questo componente supporta tutte le piattaforme e accetta un `Message` parametro che AWS IoT Greengrass passa come argomento allo script Python. Questa è la ricetta per il componente Hello World del tutorial Getting started.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.HelloWorld",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "My first AWS IoT Greengrass component.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "Message": "world"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "run": "python3 -u {artifacts:path}/hello_world.py {configuration:/Message}"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
```

```
      "Lifecycle": {
        "run": "py -3 -u {artifacts:path}/hello_world.py {configuration:/Message}"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.HelloWorld
ComponentVersion: '1.0.0'
ComponentDescription: My first AWS IoT Greengrass component.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    Message: world
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      run: |
        python3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
  - Platform:
      os: windows
    Lifecycle:
      run: |
        py -3 -u {artifacts:path}/hello_world.py "{configuration:/Message}"
```

Esempio di componente runtime Python

La seguente ricetta descrive un componente che installa Python. Questo componente supporta dispositivi Linux a 64 bit.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PythonRuntime",
  "ComponentDescription": "Installs Python 3.7",
  "ComponentPublisher": "Amazon",
```

```
  "ComponentVersion": "3.7.0",
  "Manifests": [
    {
      "Platform": {
        "os": "linux",
        "architecture": "amd64"
      },
      "Lifecycle": {
        "install": "apt-get update\napt-get install python3.7"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PythonRuntime
ComponentDescription: Installs Python 3.7
ComponentPublisher: Amazon
ComponentVersion: '3.7.0'
Manifests:
  - Platform:
      os: linux
      architecture: amd64
    Lifecycle:
      install: |
        apt-get update
        apt-get install python3.7
```

Ricetta del componente che specifica diversi campi

La seguente ricetta del componente utilizza diversi campi di ricetta.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.FooService",
  "ComponentDescription": "Complete recipe for AWS IoT Greengrass components",
  "ComponentPublisher": "Amazon",
```

```
  "ComponentVersion": "1.0.0",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "TestParam": "TestValue"
    }
  },
  "ComponentDependencies": {
    "BarService": {
      "VersionRequirement": "^1.1.0",
      "DependencyType": "SOFT"
    },
    "BazService": {
      "VersionRequirement": "^2.0.0"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux",
        "architecture": "amd64"
      },
      "Lifecycle": {
        "install": {
          "Skipif": "onpath git",
          "Script": "sudo apt-get install git"
        },
        "Setenv": {
          "environment_variable1": "variable_value1",
          "environment_variable2": "variable_value2"
        }
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/hello_world.zip",
          "Unarchive": "ZIP"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/hello_world_linux.py"
        }
      ]
    },
    {
      "Lifecycle": {
        "install": {
```

```
          "Skipif": "onpath git",
          "Script": "sudo apt-get install git",
          "RequiresPrivilege": "true"
        }
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/hello_world.py"
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.FooService
ComponentDescription: Complete recipe for AWS IoT Greengrass components
ComponentPublisher: Amazon
ComponentVersion: 1.0.0
ComponentConfiguration:
  DefaultConfiguration:
    TestParam: TestValue
ComponentDependencies:
  BarService:
    VersionRequirement: ^1.1.0
    DependencyType: SOFT
  BazService:
    VersionRequirement: ^2.0.0
Manifests:
  - Platform:
      os: linux
      architecture: amd64
    Lifecycle:
      install:
        Skipif: onpath git
        Script: sudo apt-get install git
      Setenv:
        environment_variable1: variable_value1
        environment_variable2: variable_value2
    Artifacts:
```

```
      - URI: 's3://DOC-EXAMPLE-BUCKET/hello_world.zip'
        Unarchive: ZIP
      - URI: 's3://DOC-EXAMPLE-BUCKET/hello_world_linux.py'
  - Lifecycle:
      install:
        Skipif: onpath git
        Script: sudo apt-get install git
        RequiresPrivilege: 'true'
    Artifacts:
      - URI: 's3://DOC-EXAMPLE-BUCKET/hello_world.py'
```

Riferimento alla variabile di ambiente

Il softwareAWS IoT Greengrass Core imposta le variabili di ambiente quando esegue gli script del ciclo di vita dei componenti. Puoi inserire queste variabili di ambiente nei tuoi componenti per ottenere il nome dell'oggetto e la versione del nucleo di Greengrass.Regione AWS Il software imposta anche le variabili di ambiente necessarie al componente per utilizzare l'SDK di comunicazione tra processi e interagire conAWS i servizi.

Puoi anche impostare variabili di ambiente personalizzate per gli script del ciclo di vita del tuo componente. Per ulteriori informazioni, consulta Setenv.

Il softwareAWS IoT Greengrass Core imposta le seguenti variabili di ambiente:

`AWS_IOT_THING_NAME`

Il nome dellaAWS IoT cosa che rappresenta questo dispositivo principale Greengrass.

`AWS_REGION`

IlRegione AWS luogo in cui opera questo dispositivo principale di Greengrass.

GliAWS SDK utilizzano questa variabile di ambiente per identificare la regione predefinita da utilizzare. Questa variabile è equivalente a`AWS_DEFAULT_REGION`.

`AWS_DEFAULT_REGION`

IlRegione AWS luogo in cui opera questo dispositivo principale di Greengrass.

AWS CLIUtilizza questa variabile di ambiente per identificare la regione predefinita da utilizzare. Questa variabile è equivalente a`AWS_REGION`.

`GGC_VERSION`

La versione del componente Greengrass nucleus che funziona su questo dispositivo core Greengrass.

`GG_ROOT_CA_PATH`

Questa funzionalità è disponibile per la versione 2.5.5 e successive del componente nucleus Greengrass.

Il percorso verso il certificato dell'autorità di certificazione principale (CA) utilizzato dal nucleo di Greengrass.

`AWS_GG_NUCLEUS_DOMAIN_SOCKET_FILEPATH_FOR_COMPONENT`

Il percorso verso il socket IPC utilizzato dai componenti per comunicare con il softwareAWS IoT Greengrass Core. Per ulteriori informazioni, consulta Usa il SDK per dispositivi AWS IoT per comunicare con il nucleo Greengrass, altri componenti e AWS IoT Core.

`SVCUID`

Il token segreto utilizzato dai componenti per connettersi al socket IPC e comunicare con il softwareAWS IoT Greengrass Core. Per ulteriori informazioni, consulta Usa il SDK per dispositivi AWS IoT per comunicare con il nucleo Greengrass, altri componenti e AWS IoT Core.

`AWS_CONTAINER_AUTHORIZATION_TOKEN`

Il token segreto utilizzato dai componenti per recuperare le credenziali dal componente del servizio di scambio di token.

`AWS_CONTAINER_CREDENTIALS_FULL_URI`

L'URI richiesto dai componenti per recuperare le credenziali dal componente del servizio di scambio di token.

Implementazione AWS IoT Greengrass dei componenti sui dispositivi

Puoi utilizzarli AWS IoT Greengrass per distribuire componenti su dispositivi o gruppi di dispositivi. Le distribuzioni vengono utilizzate per definire i componenti e le configurazioni che vengono inviati ai dispositivi. AWS IoT Greengrasssi distribuisce su obiettivi, AWS IoT oggetti o gruppi di oggetti che

rappresentano i dispositivi principali di Greengrass. AWS IoT Greengrassutilizza i AWS IoT Corejob per l'implementazione sui dispositivi principali. Puoi configurare la modalità di distribuzione del lavoro sui tuoi dispositivi.

Implementazioni principali dei dispositivi

Ogni dispositivo principale esegue i componenti delle distribuzioni per quel dispositivo. Una nuova distribuzione sulla stessa destinazione sovrascrive la distribuzione precedente sulla destinazione. Quando si crea una distribuzione, si definiscono i componenti e le configurazioni da applicare al software esistente del dispositivo principale.

Quando si modifica una distribuzione per una destinazione, si sostituiscono i componenti della revisione precedente con i componenti della nuova revisione. Ad esempio, si distribuiscono i Gestore segreto componenti Gestore dei registri and nel gruppo di oggetti. `TestGroup` Quindi si crea un'altra distribuzione `TestGroup` che specifica solo il componente secret manager. Di conseguenza, i dispositivi principali di quel gruppo non eseguono più il gestore dei registri.

Risoluzione delle dipendenze dalla piattaforma

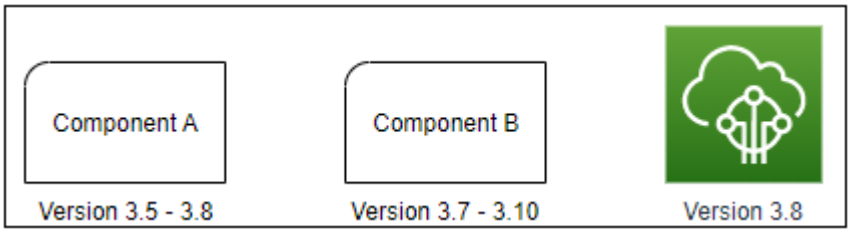
Quando un dispositivo principale riceve un'implementazione, verifica che i componenti siano compatibili con il dispositivo principale. Ad esempio, se si distribuisce il Firehose su una destinazione Windows, la distribuzione avrà esito negativo.

Risoluzione delle dipendenze dei componenti

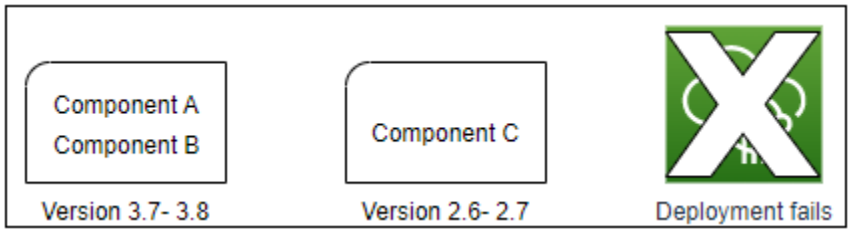
Il dispositivo principale verifica inoltre se le dipendenze di ciascun componente sono compatibili con i vincoli di versione per le distribuzioni di altri componenti in questo gruppo di oggetti. Laddove i vincoli di versione per un componente si sovrappongono, Greengrass utilizza la versione più alta applicabile del componente. Per esempio:

- Il componente A viene distribuito su. `TestGroup` Il componente A dipende dalle `com.example.PythonRuntime` versioni del componente 3.5 - 3.10.
- Quindi si distribuisce il componente B su. `TestGroup` Il componente B dipende dalle `com.example.PythonRuntime` versioni del componente da 3.7 a 3.8.

Di conseguenza, i dispositivi principali `TestGroup` stabiliscono di poter implementare la versione 3.8 del `com.example.PythonRuntime` componente, poiché questa è la versione più applicabile in cui i vincoli di versione si sovrappongono.



Quindi si distribuisce il componente C su. `TestGroup` Il componente C dipende dalle `com.example.PythonRuntime` versioni del componente 2.6 - 2.7. Questa distribuzione non riesce perché non esiste una versione del componente che soddisfi i vincoli 2.6 - 2.7 e 3.7 - 3.8.



Rimozione di un dispositivo da un gruppo di oggetti

Quando si rimuove un dispositivo principale da un gruppo di oggetti, il comportamento di distribuzione dei componenti dipende dalla versione del nucleo Greengrass utilizzata dal dispositivo principale.

2.5.1 and later

Quando si rimuove un dispositivo principale da un gruppo di oggetti, il comportamento dipende dal fatto che la AWS IoT policy conceda o meno l'autorizzazione. `greengrass:ListThingGroupsForCoreDevice` Per ulteriori informazioni su questa autorizzazione e sulle AWS IoT politiche per i dispositivi principali, consultaAutenticazione e autorizzazione del dispositivo per AWS IoT Greengrass.

- Se la AWS IoT politica concede questa autorizzazione

  Quando rimuovi un dispositivo principale da un gruppo di oggetti, AWS IoT Greengrass rimuove i componenti del gruppo di oggetti la prossima volta che viene effettuata una distribuzione sul dispositivo. Se un componente del dispositivo è incluso nella distribuzione successiva, tale componente non viene rimosso dal dispositivo.
- Se la AWS IoT politica non concede questa autorizzazione

  Quando rimuovi un dispositivo principale da un gruppo di oggetti, AWS IoT Greengrass non elimina i componenti di quel gruppo di oggetti dal dispositivo.

Per rimuovere un componente da un dispositivo, utilizzate il comando deployment create della CLI di Greengrass. Specificate il componente da rimuovere con l'`--remove`argomento e specificate il gruppo di cose con l'`--groupId`argomento.

2.5.0

Quando rimuovete un dispositivo principale da un gruppo di oggetti, AWS IoT Greengrass rimuove i componenti del gruppo di oggetti la prossima volta che viene effettuata una distribuzione sul dispositivo. Se un componente del dispositivo è incluso nella distribuzione successiva, tale componente non viene rimosso dal dispositivo.

Questo comportamento richiede che la AWS IoT politica del dispositivo principale conceda l'`greengrass:ListThingGroupsForCoreDevice`autorizzazione. Se un dispositivo principale non dispone di questa autorizzazione, il dispositivo principale non riesce ad applicare le distribuzioni. Per ulteriori informazioni, consulta Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass.

2.0.x - 2.4.x

Quando rimuovi un dispositivo principale da un gruppo di oggetti, AWS IoT Greengrass non elimina i componenti di quel gruppo di oggetti dal dispositivo.

Per rimuovere un componente da un dispositivo, utilizzate il comando deployment create della CLI di Greengrass. Specificate il componente da rimuovere con l'`--remove`argomento e specificate il gruppo di cose con l'`--groupId`argomento.

Distribuzioni

Le distribuzioni sono continue. Quando crei una distribuzione, AWS IoT Greengrass distribuisci la distribuzione sui dispositivi di destinazione che sono online. Se un dispositivo di destinazione non è online, riceve la distribuzione la volta successiva a cui si connetteAWS IoT Greengrass. Quando aggiungi un dispositivo principale a un gruppo di oggetti di destinazione, AWS IoT Greengrass invia al dispositivo la distribuzione più recente per quel gruppo di oggetti.

Prima che un dispositivo principale distribuisca un componente, per impostazione predefinita invia una notifica a ciascun componente del dispositivo. I componenti Greengrass possono rispondere alla notifica per posticipare l'implementazione. Potresti voler posticipare l'installazione se il dispositivo ha un livello di batteria basso o sta eseguendo un processo che non può essere interrotto. Per ulteriori

informazioni, consulta Tutorial: Sviluppa un componente Greengrass che rinvii gli aggiornamenti dei componenti. Quando si crea una distribuzione, è possibile configurarla per la distribuzione senza notificare i componenti.

Ogni oggetto o gruppo di oggetti di destinazione può avere una distribuzione alla volta. Ciò significa che quando si crea una distribuzione per una destinazione, AWS IoT Greengrass non viene più distribuita la revisione precedente della distribuzione di quella destinazione.

Opzioni di implementazione

Le distribuzioni offrono diverse opzioni che consentono di controllare quali dispositivi ricevono un aggiornamento e come viene distribuito l'aggiornamento. Quando si crea una distribuzione, è possibile configurare le seguenti opzioni:

- AWS IoT Greengrasscomponenti

  Definire i componenti da installare ed eseguire sui dispositivi di destinazione. AWS IoT Greengrassi componenti sono moduli software che vengono distribuiti ed eseguiti sui dispositivi core Greengrass. I dispositivi ricevono componenti solo se il componente supporta la piattaforma del dispositivo. Ciò consente di eseguire la distribuzione su gruppi di dispositivi anche se i dispositivi di destinazione funzionano su più piattaforme. Se un componente non supporta la piattaforma del dispositivo, il componente non viene distribuito sul dispositivo.

  Puoi distribuire componenti personalizzati e componenti AWS forniti sui tuoi dispositivi. Quando si distribuisce un componente, AWS IoT Greengrass identifica tutte le dipendenze tra i componenti e distribuisce anche tali componenti. Per ulteriori informazioni, consultare Sviluppa AWS IoT Greengrass componenti e AWS-componenti forniti.

  L'utente definisce la versione e l'aggiornamento della configurazione da distribuire per ogni componente. L'aggiornamento della configurazione specifica come modificare la configurazione esistente del componente sul dispositivo principale o la configurazione predefinita del componente se il componente non esiste sul dispositivo principale. È possibile specificare quali valori di configurazione ripristinare ai valori predefiniti e i nuovi valori di configurazione da unire al dispositivo principale. Quando un dispositivo principale riceve distribuzioni per destinazioni diverse e ogni distribuzione specifica versioni dei componenti compatibili, il dispositivo principale applica gli aggiornamenti di configurazione in base al timestamp di quando si crea la distribuzione. Per ulteriori informazioni, consulta Aggiornamento delle configurazioni dei componenti.

> **Important**
>
> Quando distribuisci un componente, AWS IoT Greengrass installa le ultime versioni supportate di tutte le dipendenze di quel componente. Per questo motivo, le nuove versioni patch dei componenti pubblici AWS forniti potrebbero essere distribuite automaticamente sui dispositivi principali se si aggiungono nuovi dispositivi a un gruppo di oggetti o si aggiorna la distribuzione destinata a tali dispositivi. Alcuni aggiornamenti automatici, come un aggiornamento Nucleus, possono causare il riavvio imprevisto dei dispositivi. Per evitare aggiornamenti involontari per un componente in esecuzione sul tuo dispositivo, ti consigliamo di includere direttamente la versione preferita di quel componente quando crei una distribuzione. Per ulteriori informazioni sul comportamento di aggiornamento per il software AWS IoT Greengrass Core, consultaAggiornamento del software AWS IoT Greengrass Core (OTA).

- Politiche di distribuzione

  Definisci quando è sicuro implementare una configurazione e cosa fare se l'implementazione fallisce. Puoi specificare se attendere o meno che i componenti segnalino che possono essere aggiornati. È inoltre possibile specificare se ripristinare o meno i dispositivi alla configurazione precedente se applicano una distribuzione che non riesce.

- Interrompi la configurazione

  Definisci quando e come interrompere una distribuzione. La distribuzione si interrompe e fallisce se vengono soddisfatti i criteri definiti. Ad esempio, è possibile configurare una distribuzione in modo che si interrompa se una percentuale di dispositivi non riesce ad applicarla dopo che un numero minimo di dispositivi l'ha ricevuta.

- Rollout configuration (Configurazione rollout)

  Definisci la velocità con cui una distribuzione viene distribuita ai dispositivi di destinazione. È possibile configurare un aumento esponenziale della velocità con limiti di velocità minimi e massimi.

- Configurazione del timeout

  Definisci la quantità massima di tempo a disposizione di ciascun dispositivo per applicare una distribuzione. Se un dispositivo supera la durata specificata, il dispositivo non riesce ad applicare la distribuzione.

> **Important**
>
> I componenti personalizzati possono definire artefatti nei bucket S3. Quando il software AWS IoT Greengrass Core distribuisce un componente, scarica gli artefatti del componente da. Cloud AWS Per impostazione predefinita, i ruoli principali dei dispositivi non consentono l'accesso ai bucket S3. Per distribuire componenti personalizzati che definiscono gli artefatti in un bucket S3, il ruolo principale del dispositivo deve concedere le autorizzazioni per scaricare gli artefatti da quel bucket. Per ulteriori informazioni, consulta Consenti l'accesso ai bucket S3 per gli artefatti dei componenti.

Argomenti

Creare distribuzione

È possibile creare una distribuzione destinata a un oggetto o a un gruppo di oggetti.

Quando si crea una distribuzione, si configurano i componenti software da distribuire e il modo in cui il processo di distribuzione viene distribuito sui dispositivi di destinazione. È possibile definire la distribuzione nel file JSON fornito a. AWS CLI

L'obiettivo di distribuzione determina i dispositivi su cui si desidera eseguire i componenti. Per eseguire la distribuzione su un dispositivo principale, specifica un elemento. Per eseguire la distribuzione su più dispositivi principali, specificate un gruppo di oggetti che includa tali dispositivi. Per ulteriori informazioni su come configurare i gruppi di oggetti, vedere Thing group statici e Thing group dinamici nella AWS IoTDeveloper Guide.

Segui i passaggi di questa sezione per creare una distribuzione su un target. Per ulteriori informazioni su come aggiornare i componenti software su una destinazione con una distribuzione, consultaRivedi le distribuzioni.

> **Warning**
>
> L'CreateDeploymentoperazione può disinstallare componenti dai dispositivi principali. Se un componente è presente nella distribuzione precedente e non in quella nuova, il dispositivo principale disinstalla quel componente. Per evitare la disinstallazione dei componenti, utilizza innanzitutto l'ListDeploymentsoperazione per verificare se la destinazione per la distribuzione ha già una distribuzione esistente. Quindi, utilizza l'GetDeploymentoperazione per iniziare da quella distribuzione esistente quando ne crei una nuova.

Per creare una distribuzione (AWS CLI)

1. Create un file chiamato`deployment.json`, quindi copiate il seguente oggetto JSON nel file. Sostituisci *`targetARN`* con l'ARN dell'oggetto o del gruppo AWS IoT di oggetti da utilizzare come target per la distribuzione. Gli ARN degli oggetti e dei gruppi di oggetti hanno il seguente formato:

   - Oggetto: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`thingName`*
   - Gruppo di oggetti: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*

   ```
   {
     "targetArn": "targetArn"
   }
   ```

2. Verifica se il target di distribuzione ha una distribuzione esistente che desideri modificare. Esegui questa operazione:

   a. Esegui il comando seguente per elencare le distribuzioni per l'obiettivo di distribuzione. Sostituisci *`targetARN`* con l'ARN dell'oggetto o del gruppo di oggetti di destinazione. AWS IoT

      ```
      aws greengrassv2 list-deployments --target-arn targetArn
      ```

      La risposta contiene un elenco con la distribuzione più recente per l'obiettivo. Se la risposta è vuota, la destinazione non ha una distribuzione esistente e puoi saltare aStep 3. Altrimenti, copia il `deploymentId` codice dalla risposta da utilizzare nel passaggio successivo.

> **Note**
>
> Puoi anche modificare una distribuzione diversa dalla revisione più recente per la destinazione. Specificare l'`--history-filter ALL`argomento per elencare tutte le distribuzioni per l'obiettivo. Quindi, copia l'ID della distribuzione che desideri modificare.

b. Esegui il comando seguente per ottenere i dettagli della distribuzione. Questi dettagli includono metadati, componenti e configurazione del processo. Sostituisci *`DeploymentID`* con l'ID del passaggio precedente.

```
aws greengrassv2 get-deployment --deployment-id deploymentId
```

La risposta contiene i dettagli della distribuzione.

c. Copia una delle seguenti coppie chiave-valore dalla risposta del comando precedente in. `deployment.json` È possibile modificare questi valori per la nuova distribuzione.

- `deploymentName`— Il nome della distribuzione.
- `components`— I componenti della distribuzione. Per disinstallare un componente, rimuovilo da questo oggetto.
- `deploymentPolicies`— Le politiche della distribuzione.
- `iotJobConfiguration`— La configurazione del lavoro della distribuzione.
- `tags`— I tag della distribuzione.

3. (Facoltativo) Definire un nome per la distribuzione. Sostituisci *`DeploymentName`* con il nome della distribuzione.

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName"
}
```

4. Aggiungi ogni componente per distribuire i dispositivi di destinazione. A tale scopo, aggiungete coppie chiave-valore all'`components`oggetto, dove la chiave è il nome del componente e il valore è un oggetto che contiene i dettagli di quel componente. Specificate i seguenti dettagli per ogni componente che aggiungete:

- `version`— La versione del componente da distribuire.
- `configurationUpdate`— L'aggiornamento della configurazione da distribuire. L'aggiornamento è un'operazione di patch che modifica la configurazione esistente del componente su ciascun dispositivo di destinazione o la configurazione predefinita del componente se non esiste sul dispositivo di destinazione. È possibile specificare i seguenti aggiornamenti di configurazione:
  - Reset updates (`reset`) — (Facoltativo) Un elenco di puntatori JSON che definiscono i valori di configurazione da ripristinare ai valori predefiniti sul dispositivo di destinazione. Il software AWS IoT Greengrass Core esegue la reimpostazione degli aggiornamenti prima di eseguire l'unione degli aggiornamenti. Per ulteriori informazioni, consulta Reimposta gli aggiornamenti.
  - Merge updates (`merge`) - (Facoltativo) Un documento JSON che definisce i valori di configurazione da unire al dispositivo di destinazione. È necessario serializzare il documento JSON come stringa. Per ulteriori informazioni, consulta Unisci gli aggiornamenti.
- `runWith`— (Facoltativo) Le opzioni di processo del sistema utilizzate dal software AWS IoT Greengrass Core per eseguire i processi di questo componente sul dispositivo principale. Se omettete un parametro nell'`runWith`oggetto, il software AWS IoT Greengrass Core utilizza i valori predefiniti configurati sul componente Greengrass nucleus.

  È possibile specificare una delle seguenti opzioni:
  - `posixUser`— L'utente e, facoltativamente, il gruppo del sistema POSIX da utilizzare per eseguire questo componente sui dispositivi core Linux. L'utente e il gruppo, se specificato, devono esistere su ogni dispositivo principale Linux. Specifica l'utente e il gruppo separati da due punti (`:`) nel seguente formato: `user:group`. Il gruppo è facoltativo. Se non si specifica un gruppo, il software AWS IoT Greengrass Core utilizza il gruppo primario per l'utente. Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti.
  - `windowsUser`— L'utente Windows da utilizzare per eseguire questo componente sui dispositivi Windows principali. L'utente deve esistere su ogni dispositivo Windows principale e il nome e la password devono essere memorizzati nell'istanza di Credentials Manager dell' LocalSystem account. Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti.

    Questa funzionalità è disponibile per la versione 2.5.0 e successive del componente Greengrass nucleus.

- `systemResourceLimits`— I limiti delle risorse di sistema da applicare ai processi di questo componente. È possibile applicare limiti di risorse di sistema a componenti Lambda generici e non containerizzati. Per ulteriori informazioni, consulta Configura i limiti delle risorse di sistema per i componenti.

  È possibile specificare una delle seguenti opzioni:

  - `cpus`— La quantità massima di tempo di CPU che i processi di questo componente possono utilizzare sul dispositivo principale. Il tempo totale della CPU di un dispositivo principale è equivalente al numero di core CPU del dispositivo. Ad esempio, su un dispositivo principale con 4 core CPU, è possibile impostare questo valore in modo da 2 limitare i processi di questo componente al 50% di utilizzo di ciascun core della CPU. Su un dispositivo con 1 core di CPU, puoi impostare questo valore `0.25` per limitare i processi di questo componente al 25 percento di utilizzo della CPU. Se imposti questo valore su un numero maggiore del numero di core della CPU, il software AWS IoT Greengrass Core non limita l'utilizzo della CPU del componente.
  - `memory`— La quantità massima di RAM (in kilobyte) che i processi di questo componente possono utilizzare sul dispositivo principale.

  Questa funzionalità è disponibile per la versione 2.4.0 e successive del componente Greengrass nucleus. AWS IoT Greengrassattualmente non supporta questa funzionalità sui dispositivi Windows core.

Example Esempio di aggiornamento della configurazione di base

L'`components`oggetto di esempio seguente specifica di distribuire un componente`com.example.PythonRuntime`, che prevede un parametro di configurazione denominato. `pythonVersion`

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName",
  "components": {
    "com.example.PythonRuntime": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"pythonVersion\":\"3.7\"}"
```

```
      }
    }
  }
}
```

Example Esempio di aggiornamento della configurazione con ripristino e unione degli aggiornamenti

Consideriamo un esempio di componente di dashboard industriale con la seguente configurazione predefinita. `com.example.IndustrialDashboard`

```
{
  "name": null,
  "mode": "REQUEST",
  "network": {
    "useHttps": true,
    "port": {
      "http": 80,
      "https": 443
    },
  },
  "tags": []
}
```

Il seguente aggiornamento della configurazione specifica le seguenti istruzioni:

1. Reimposta l'impostazione HTTPS al valore predefinito (`true`).
2. Reimposta l'elenco dei tag industriali su un elenco vuoto.
3. Unisci un elenco di etichette industriali che identificano i flussi di dati di temperatura e pressione per due caldaie.

```
{
  "reset": [
    "/network/useHttps",
    "/tags"
  ],
  "merge": {
    "tags": [
      "/boiler/1/temperature",
      "/boiler/1/pressure",
```

```
      "/boiler/2/temperature",
      "/boiler/2/pressure"
    ]
  }
}
```

L'`componentsoggetto` di esempio seguente specifica la distribuzione di questo componente del dashboard industriale e l'aggiornamento della configurazione.

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "reset": [
          "/network/useHttps",
          "/tags"
        ],
        "merge": "{\"tags\":[\"/boiler/1/temperature\",\"/boiler/1/pressure\",\"/
boiler/2/temperature\",\"/boiler/2/pressure\"]}"
      }
    }
  }
}
```

5. (Facoltativo) Definire le politiche di distribuzione per la distribuzione. È possibile configurare quando i dispositivi principali possono applicare una distribuzione in sicurezza o cosa fare se un dispositivo principale non riesce ad applicare la distribuzione. A tale scopo, aggiungi un `deploymentPolicies` oggetto e quindi esegui una delle seguenti operazioni: `deployment.json`

   1. (Facoltativo) Specificate la politica di aggiornamento dei componenti (`componentUpdatePolicy`). Questa politica definisce se la distribuzione consente o meno ai componenti di posticipare un aggiornamento finché non sono pronti per l'aggiornamento. Ad esempio, potrebbe essere necessario ripulire le risorse o completare azioni critiche prima di poter riavviarsi per applicare un aggiornamento. Questa politica definisce anche la quantità di tempo a disposizione dei componenti per rispondere a una notifica di aggiornamento.

      Questa politica è un oggetto con i seguenti parametri:

- `action`— (Facoltativo) Se notificare o meno i componenti e attendere che vengano segnalati quando sono pronti per l'aggiornamento. Seleziona una delle opzioni seguenti:
  - `NOTIFY_COMPONENTS`: l'implementazione avvisa ogni componente prima che venga interrotto e aggiornato. I componenti possono utilizzare l'operazione SubscribeToComponentUpdates IPC per ricevere queste notifiche.
  - `SKIP_NOTIFY_COMPONENTS`: l'implementazione non avvisa i componenti né attende che vengano aggiornati in sicurezza.

  L'impostazione predefinita è `NOTIFY_COMPONENTS`.
- `timeoutInSecondsLa` quantità di tempo in secondi a disposizione di ciascun componente per rispondere a una notifica di aggiornamento con l'operazione DeferComponentUpdate IPC. Se il componente non risponde entro questo lasso di tempo, la distribuzione procede sul dispositivo principale.

  Il valore predefinito è 60 secondi.

2. (Facoltativo) Specificare la politica di convalida della configurazione (). `configurationValidationPolicy` Questa politica definisce il tempo a disposizione di ciascun componente per convalidare un aggiornamento della configurazione da una distribuzione. I componenti possono utilizzare l'operazione SubscribeToValidateConfigurationUpdates IPC per sottoscrivere le notifiche relative ai propri aggiornamenti di configurazione. Quindi, i componenti possono utilizzare l'operazione SendConfigurationValidityReport IPC per comunicare al software AWS IoT Greengrass Core se l'aggiornamento della configurazione è valido. Se l'aggiornamento della configurazione non è valido, la distribuzione non riesce.

   Questa politica è un oggetto con il seguente parametro:

   - `timeoutInSeconds`(Facoltativo) La quantità di tempo in secondi a disposizione di ogni componente per convalidare un aggiornamento della configurazione. Se il componente non risponde entro questo lasso di tempo, la distribuzione procede sul dispositivo principale.

     Il valore predefinito è 30 secondi.

3. (Facoltativo) Specificare la politica di gestione degli errori ()`failureHandlingPolicy`. Questa policy è una stringa che definisce se ripristinare o meno i dispositivi se la distribuzione fallisce. Seleziona una delle opzioni seguenti:

   - `ROLLBACK`— Se l'installazione fallisce su un dispositivo principale, il software AWS IoT Greengrass Core ripristina il dispositivo principale alla configurazione precedente.

- `DO_NOTHING`— Se l'installazione fallisce su un dispositivo principale, il software AWS IoT Greengrass Core mantiene la nuova configurazione. Ciò può causare il danneggiamento dei componenti se la nuova configurazione non è valida.

L'impostazione predefinita è `ROLLBACK`.

La tua distribuzione in `deployment.json` potrebbe essere simile al seguente esempio:

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "reset": [
          "/network/useHttps",
          "/tags"
        ],
        "merge": "{\"tags\":[\"/boiler/1/temperature\",\"/boiler/1/pressure\",\"/
boiler/2/temperature\",\"/boiler/2/pressure\"]}"
      }
    }
  },
  "deploymentPolicies": {
    "componentUpdatePolicy": {
      "action": "NOTIFY_COMPONENTS",
      "timeoutInSeconds": 30
    },
    "configurationValidationPolicy": {
      "timeoutInSeconds": 60
    },
    "failureHandlingPolicy": "ROLLBACK"
  }
}
```

6. (Facoltativo) Definisci in che modo la distribuzione viene interrotta, implementata o scaduta. AWS IoT Greengrassutilizza i AWS IoT Core processi per inviare le distribuzioni ai dispositivi principali, quindi queste opzioni sono identiche alle opzioni di configurazione per AWS IoT Core i lavori. Per ulteriori informazioni, consulta Job rollout and abort configuration nella AWS IoTDeveloper Guide.

Per definire le opzioni di lavoro, aggiungi un `iotJobConfiguration` oggetto a. `deployment.json` Quindi, definite le opzioni da configurare.

La tua distribuzione in `deployment.json` potrebbe essere simile all'esempio seguente:

```
{
  "targetArn": "targetArn",
  "deploymentName": "deploymentName",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "reset": [
          "/network/useHttps",
          "/tags"
        ],
        "merge": "{\"tags\":[\"/boiler/1/temperature\",\"/boiler/1/pressure\",\"/
boiler/2/temperature\",\"/boiler/2/pressure\"]}"
      }
    }
  },
  "deploymentPolicies": {
    "componentUpdatePolicy": {
      "action": "NOTIFY_COMPONENTS",
      "timeoutInSeconds": 30
    },
    "configurationValidationPolicy": {
      "timeoutInSeconds": 60
    },
    "failureHandlingPolicy": "ROLLBACK"
  },
  "iotJobConfiguration": {
    "abortConfig": {
      "criteriaList": [
        {
          "action": "CANCEL",
          "failureType": "ALL",
          "minNumberOfExecutedThings": 100,
          "thresholdPercentage": 5
        }
      ]
    },
```

```
    "jobExecutionsRolloutConfig": {
      "exponentialRate": {
        "baseRatePerMinute": 5,
        "incrementFactor": 2,
        "rateIncreaseCriteria": {
          "numberOfNotifiedThings": 10,
          "numberOfSucceededThings": 5
        }
      },
      "maximumPerMinute": 50
    },
    "timeoutConfig":  {
      "inProgressTimeoutInMinutes": 5
    }
  }
}
```

7. (Facoltativo) Aggiungi tag (`tags`) per la distribuzione. Per ulteriori informazioni, consulta Tagging delle risorse AWS IoT Greengrass Version 2..
8. Esegui il comando seguente per creare la distribuzione da`deployment.json`.

   ```
   aws greengrassv2 create-deployment --cli-input-json file://deployment.json
   ```

   La risposta include un messaggio `deploymentId` che identifica questa distribuzione. È possibile utilizzare l'ID di distribuzione per verificare lo stato della distribuzione. Per ulteriori informazioni, consulta Controllo dello stato di implementazione.

Aggiornamento delle configurazioni dei componenti

Le configurazioni dei componenti sono oggetti JSON che definiscono i parametri per ogni componente. La ricetta di ogni componente definisce la sua configurazione predefinita, che viene modificata quando si distribuiscono i componenti sui dispositivi principali.

Quando si crea una distribuzione, è possibile specificare l'aggiornamento della configurazione da applicare a ciascun componente. Gli aggiornamenti di configurazione sono operazioni di patch, il che significa che l'aggiornamento modifica la configurazione del componente esistente sul dispositivo principale. Se il dispositivo principale non dispone del componente, l'aggiornamento della configurazione modifica e applica la configurazione predefinita per quella distribuzione.

L'aggiornamento della configurazione definisce gli aggiornamenti di ripristino e gli aggiornamenti di unione. Gli aggiornamenti di ripristino definiscono quali valori di configurazione ripristinare i valori predefiniti o rimuovere. Gli aggiornamenti di fusione definiscono i nuovi valori di configurazione da impostare per il componente. Quando si distribuisce un aggiornamento di configurazione, il software AWS IoT Greengrass Core esegue l'aggiornamento di ripristino prima dell'aggiornamento di fusione.

I componenti possono convalidare gli aggiornamenti di configurazione distribuiti. Il componente si iscrive per ricevere una notifica quando una distribuzione modifica la sua configurazione e può rifiutare una configurazione che non supporta. Per ulteriori informazioni, consulta Interazione con la configurazione dei componenti.

Argomenti

- Reimposta gli aggiornamenti
- Unisci gli aggiornamenti
- Esempi

Reimposta gli aggiornamenti

Gli aggiornamenti di ripristino definiscono quali valori di configurazione ripristinare ai valori predefiniti sul dispositivo principale. Se un valore di configurazione non ha un valore predefinito, l'aggiornamento di ripristino rimuove tale valore dalla configurazione del componente. Questo può aiutarti a correggere un componente che si rompe a causa di una configurazione non valida.

Usa un elenco di puntatori JSON per definire i valori di configurazione da reimpostare. I puntatori JSON iniziano con una barra. / Per identificare un valore in una configurazione di componenti annidati, utilizzate forward slashes (/) per separare le chiavi per ogni livello della configurazione. Per ulteriori informazioni, vedete la specifica del puntatore JSON.

> **Note**
>
> È possibile ripristinare solo i valori predefiniti di un intero elenco. Non è possibile utilizzare la reimpostazione degli aggiornamenti per reimpostare un singolo elemento in un elenco.

Per ripristinare l'intera configurazione di un componente ai valori predefiniti, specifica una singola stringa vuota come aggiornamento di ripristino.

```
"reset": [""]
```

Unisci gli aggiornamenti

Gli aggiornamenti di fusione definiscono i valori di configurazione da inserire nella configurazione del componente sul core. L'aggiornamento di fusione è un oggetto JSON che il software AWS IoT Greengrass Core unisce dopo aver ripristinato i valori nei percorsi specificati nell'aggiornamento di ripristino. Quando utilizzate gli AWS CLI o AWS SDK, dovete serializzare questo oggetto JSON come stringa.

Puoi unire una coppia chiave-valore che non esiste nella configurazione predefinita del componente. È inoltre possibile unire una coppia chiave-valore di tipo diverso dal valore con la stessa chiave. Il nuovo valore sostituisce il vecchio valore. Ciò significa che è possibile modificare la struttura dell'oggetto di configurazione.

È possibile unire valori nulli e stringhe, elenchi e oggetti vuoti.

> **Note**
>
> Non è possibile utilizzare gli aggiornamenti di fusione allo scopo di inserire o aggiungere un elemento a un elenco. È possibile sostituire un intero elenco oppure definire un oggetto in cui ogni elemento ha una chiave unica.
> AWS IoT Greengrass utilizza JSON per i valori di configurazione. JSON specifica un tipo di numero ma non distingue tra numeri interi e float. Di conseguenza, i valori di configurazione potrebbero essere convertiti in float in. AWS IoT Greengrass Per garantire che il componente utilizzi il tipo di dati corretto, si consiglia di definire i valori di configurazione numerici come stringhe. Quindi, chiedi al componente di analizzarli come numeri interi o float. Ciò garantisce che i valori di configurazione abbiano lo stesso tipo nella configurazione e sul dispositivo principale.

Usa le variabili di ricetta negli aggiornamenti di fusione

Questa funzionalità è disponibile per la versione 2.6.0 e successive del componente Greengrass nucleus.

Se imposti l'opzione di interpolateComponentConfigurationconfigurazione Greengrass nucleus su`true`, puoi utilizzare variabili di ricetta, diverse dalla variabile *`component_dependency_name`*`:configuration:`*`json_pointer`* recipe, negli aggiornamenti di fusione. Ad esempio, è possibile utilizzare la variabile `{iot:thingName}` recipe in un aggiornamento di fusione per includere il nome dell' AWS IoT oggetto del dispositivo principale

in un valore di configurazione del componente, ad esempio una politica di autorizzazione per la comunicazione tra processi (IPC).

Esempi

L'esempio seguente mostra gli aggiornamenti di configurazione per un componente del dashboard con la seguente configurazione predefinita. Questo componente di esempio visualizza informazioni sulle apparecchiature industriali.

```
{
  "name": null,
  "mode": "REQUEST",
  "network": {
    "useHttps": true,
    "port": {
      "http": 80,
      "https": 443
    },
  },
  "tags": []
}
```

Ricetta dei componenti del cruscotto industriale

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.IndustrialDashboard",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Displays information about industrial equipment.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "name": null,
      "mode": "REQUEST",
      "network": {
        "useHttps": true,
        "port": {
          "http": 80,
          "https": 443
        },
      },
```

```
      "tags": []
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "run": "python3 -u {artifacts:path}/industrial_dashboard.py"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "run": "py -3 -u {artifacts:path}/industrial_dashboard.py"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.IndustrialDashboard
ComponentVersion: '1.0.0'
ComponentDescription: Displays information about industrial equipment.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    name: null
    mode: REQUEST
    network:
      useHttps: true
      port:
        http: 80
        https: 443
    tags: []
Manifests:
  - Platform:
```

```
    os: linux
  Lifecycle:
    run: |
      python3 -u {artifacts:path}/industrial_dashboard.py
- Platform:
    os: windows
  Lifecycle:
    run: |
      py -3 -u {artifacts:path}/industrial_dashboard.py
```

Example Esempio 1: Merge update

Si crea una distribuzione che applica il seguente aggiornamento di configurazione, che specifica un aggiornamento di unione ma non un aggiornamento di ripristino. Questo aggiornamento di configurazione indica al componente di visualizzare il dashboard sulla porta HTTP 8080 con i dati provenienti da due caldaie.

Console

Configurazione da unire

```
{
  "name": "Factory 2A",
  "network": {
    "useHttps": false,
    "port": {
      "http": 8080
    }
  },
  "tags": [
    "/boiler/1/temperature",
    "/boiler/1/pressure",
    "/boiler/2/temperature",
    "/boiler/2/pressure"
  ]
}
```

AWS CLI

Il comando seguente crea una distribuzione su un dispositivo principale.

```
aws greengrassv2 create-deployment --cli-input-json file://dashboard-deployment.json
```

Il `dashboard-deployment.json` file contiene il seguente documento JSON.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"name\":\"Factory 2A\",\"network\":{\"useHttps\":false,\"port
\":{\"http\":8080}},\"tags\":[\"/boiler/1/temperature\",\"/boiler/1/pressure\",\"/
boiler/2/temperature\",\"/boiler/2/pressure\"]}"
      }
    }
  }
}
```

Greengrass CLI

Il seguente comando Greengrass CLI crea una distribuzione locale su un dispositivo principale.

```
sudo greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.IndustrialDashboard=1.0.0" \
  --update-config dashboard-configuration.json
```

Il `dashboard-configuration.json` file contiene il seguente documento JSON.

```
{
  "com.example.IndustrialDashboard": {
    "MERGE": {
      "name": "Factory 2A",
      "network": {
        "useHttps": false,
        "port": {
          "http": 8080
        }
      },
```

```
      "tags": [
        "/boiler/1/temperature",
        "/boiler/1/pressure",
        "/boiler/2/temperature",
        "/boiler/2/pressure"
      ]
    }
  }
}
```

Dopo questo aggiornamento, il componente del dashboard presenta la seguente configurazione.

```
{
  "name": "Factory 2A",
  "mode": "REQUEST",
  "network": {
    "useHttps": false,
    "port": {
      "http": 8080,
      "https": 443
    }
  },
  "tags": [
    "/boiler/1/temperature",
    "/boiler/1/pressure",
    "/boiler/2/temperature",
    "/boiler/2/pressure"
  ]
}
```

Example Esempio 2: reimpostazione e unione degli aggiornamenti

Quindi, si crea una distribuzione che applica il seguente aggiornamento di configurazione, che specifica un aggiornamento di ripristino e un aggiornamento di unione. Questi aggiornamenti specificano la visualizzazione del dashboard sulla porta HTTPS predefinita con dati provenienti da diverse caldaie. Questi aggiornamenti modificano la configurazione risultante dagli aggiornamenti di configurazione dell'esempio precedente.

Console

Reimposta i percorsi

```
[
  "/network/useHttps",
  "/tags"
]
```

Configurazione da unire

```
{
  "tags": [
    "/boiler/3/temperature",
    "/boiler/3/pressure",
    "/boiler/4/temperature",
    "/boiler/4/pressure"
  ]
}
```

AWS CLI

Il comando seguente crea una distribuzione su un dispositivo principale.

```
aws greengrassv2 create-deployment --cli-input-json file://dashboard-
deployment2.json
```

Il `dashboard-deployment2.json` file contiene il seguente documento JSON.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.IndustrialDashboard": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "reset": [
          "/network/useHttps",
          "/tags"
        ],
```

```
        "merge": "{\"tags\":[\"/boiler/3/temperature\",\"/boiler/3/pressure\",\"/
boiler/4/temperature\",\"/boiler/4/pressure\"]}"
      }
    }
  }
}
```

Greengrass CLI

Il seguente comando Greengrass CLI crea una distribuzione locale su un dispositivo principale.

```
sudo greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.IndustrialDashboard=1.0.0" \
  --update-config dashboard-configuration2.json
```

Il `dashboard-configuration2.json` file contiene il seguente documento JSON.

```
{
  "com.example.IndustrialDashboard": {
    "RESET": [
      "/network/useHttps",
      "/tags"
    ],
    "MERGE": {
      "tags": [
        "/boiler/3/temperature",
        "/boiler/3/pressure",
        "/boiler/4/temperature",
        "/boiler/4/pressure"
      ]
    }
  }
}
```

Dopo questo aggiornamento, il componente del dashboard presenta la seguente configurazione.

```
{
  "name": "Factory 2A",
  "mode": "REQUEST",
  "network": {
```

```
    "useHttps": true,
    "port": {
      "http": 8080,
      "https": 443
    }
  },
  "tags": [
    "/boiler/3/temperature",
    "/boiler/3/pressure",
    "/boiler/4/temperature",
    "/boiler/4/pressure",
  ]
}
```

Creare distribuzioni secondarie

> **Note**
>
> La funzionalità di distribuzione secondaria è disponibile su Greengrass nucleus versione 2.9.0 e successive. Non è possibile implementare una configurazione in una sottodistribuzione con versioni precedenti dei componenti di Greengrass nucleus.

Una distribuzione secondaria è una distribuzione destinata a un sottoinsieme più piccolo di dispositivi all'interno di una distribuzione principale. È possibile utilizzare le distribuzioni secondarie per distribuire una configurazione in un sottoinsieme più piccolo di dispositivi. È inoltre possibile creare distribuzioni secondarie per riprovare una distribuzione principale non riuscita in caso di guasto di uno o più dispositivi in quella distribuzione principale. Con questa funzionalità, è possibile selezionare i dispositivi che hanno avuto un errore nella distribuzione principale e creare una sottodistribuzione per testare le configurazioni fino al successo della sottodistribuzione. Una volta completata la sottodistribuzione, è possibile ridistribuire tale configurazione nella distribuzione principale.

Segui i passaggi in questa sezione per creare una sottodistribuzione e verificarne lo stato. Per ulteriori informazioni su come creare distribuzioni, consulta Creare distribuzioni.

Per creare una sottodistribuzione () AWS CLI

1. Esegui il comando seguente per recuperare le distribuzioni più recenti per un gruppo di oggetti. Sostituire l'ARN nel comando con l'ARN del gruppo di cose da interrogare. Imposta `--history-filter` **`LATEST_ONLY`** per visualizzare l'ultima implementazione di quel gruppo di cose.

```
aws greengrassv2 list-deployments --target-arn arn:aws:iot:region:account-
id:thinggroup/thingGroupName --history-filter LATEST_ONLY
```

2. Copia il file `deploymentId` dalla risposta al list-deployments comando da utilizzare nel passaggio successivo.
3. Esegui il comando seguente per recuperare lo stato di una distribuzione. Sostituisci *`deploymentId`* con l'ID della distribuzione da interrogare.

```
aws greengrassv2 get-deployment --deployment-id deploymentId
```

4. Copia il codice `iotJobId` dalla risposta al get-deployment comando da utilizzare nel passaggio successivo.
5. Esegui il comando seguente per recuperare l'elenco delle esecuzioni di job per il job specificato. Sostituisci *`JobID`* con quello del `iotJobId` passaggio precedente. Sostituisci *`lo stato`* con lo stato per cui desideri filtrare. Puoi filtrare i risultati con i seguenti stati:

   - `QUEUED`
   - `IN_PROGRESS`
   - `SUCCEEDED`
   - `FAILED`
   - `TIMED_OUT`
   - `REJECTED`
   - `REMOVED`
   - `CANCELED`

```
aws iot list-job-executions-for-job --job-id jobID --status status
```

6. Crea un nuovo gruppo di AWS IoT cose o utilizza un gruppo di cose esistente per la tua distribuzione secondaria. Quindi, aggiungi AWS IoT qualcosa a questo gruppo di oggetti. Utilizzi i gruppi di cose per gestire flotte di dispositivi core Greengrass. Quando distribuisci componenti software sui tuoi dispositivi, puoi scegliere come target singoli dispositivi o gruppi di dispositivi. È possibile aggiungere un dispositivo a un gruppo di oggetti con una distribuzione Greengrass attiva. Una volta aggiunto, è possibile distribuire i componenti software di quel gruppo di oggetti su quel dispositivo.

Per creare un nuovo gruppo di oggetti e aggiungervi i dispositivi, effettuate le seguenti operazioni:

a. Crea un gruppo AWS IoT di oggetti. Sostituire *MyGreengrassCoreGroup*con il nome del nuovo gruppo di oggetti. Non è possibile utilizzare i due punti (:)) nel nome di un gruppo di oggetti.

> **Note**
>
> Se un gruppo di oggetti per una distribuzione secondaria viene utilizzato con un`parentTargetArn`, non può essere riutilizzato con un parco dati principale diverso. Se un gruppo di oggetti è già stato utilizzato per creare una sottodistribuzione per un'altra flotta, l'API restituirà un errore.

```
aws iot create-thing-group --thing-group-name MyGreengrassCoreGroup
```

Se la richiesta ha esito positivo, la risposta è simile all'esempio seguente:

```
{
  "thingGroupName": "MyGreengrassCoreGroup",
  "thingGroupArn": "arn:aws:iot:us-
west-2:123456789012:thinggroup/MyGreengrassCoreGroup",
  "thingGroupId": "4df721e1-ff9f-4f97-92dd-02db4e3f03aa"
}
```

b. Aggiungi un core Greengrass fornito al tuo gruppo di cose. Esegui il comando seguente con questi parametri:

- Sostituiscilo *MyGreengrassCore*con il nome del core Greengrass fornito.
- *MyGreengrassCoreGroup*Sostituiscilo con il nome del tuo gruppo di oggetti.

```
aws iot add-thing-to-thing-group --thing-name MyGreengrassCore --thing-group-
name MyGreengrassCoreGroup
```

Il comando non ha alcun output se la richiesta ha esito positivo.

7. Crea un file chiamato`deployment.json`, quindi copia il seguente oggetto JSON nel file. Sostituisci *`targetARN`* con l'ARN del gruppo di oggetti da utilizzare come target per la AWS IoT distribuzione secondaria. Un obiettivo di sottodistribuzione può essere solo un gruppo di oggetti. Gli ARN dei gruppi di cose hanno il seguente formato:

   - Gruppo di cose: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thinggroup/`*`thingGroupName`*

   ```
   {
     "targetArn": "targetArn"
   }
   ```

8. Esegui nuovamente il comando seguente per ottenere i dettagli della distribuzione originale. Questi dettagli includono metadati, componenti e configurazione del processo. Sostituisci *`DeploymentID`* con l'ID di. Step 1 È possibile utilizzare questa configurazione di distribuzione per configurare la sottodistribuzione e apportare le modifiche necessarie.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

   La risposta contiene i dettagli della distribuzione. Copia una delle seguenti coppie chiave-valore dalla risposta del get-deployment comando in. `deployment.json` È possibile modificare questi valori per la distribuzione secondaria. Per ulteriori informazioni sui dettagli di questo comando, vedere GetDeployment.

   - `components`— I componenti della distribuzione. Per disinstallare un componente, rimuovilo da questo oggetto.
   - `deploymentName`— Il nome della distribuzione.
   - `deploymentPolicies`— Le politiche della distribuzione.
   - `iotJobConfiguration`— La configurazione del lavoro della distribuzione.
   - `parentTargetArn`— L'obiettivo della distribuzione principale.
   - `tags`— I tag della distribuzione.

9. Esegui il comando seguente per creare la sottodistribuzione da`deployment.json`. Sostituisci *`subDeploymentName con un nome`* per la sottodistribuzione.

   ```
   aws greengrassv2 create-deployment --deployment-name subdeploymentName --cli-input-
   json file://deployment.json
   ```

La risposta include un `deploymentId` messaggio che identifica questa sottodistribuzione. È possibile utilizzare l'ID di distribuzione per verificare lo stato della distribuzione. Per ulteriori informazioni, consulta Verifica dello stato della distribuzione.

10. Se la distribuzione secondaria ha esito positivo, è possibile utilizzare la relativa configurazione per rivedere la distribuzione principale. Copia quello `deployment.json` che hai usato nel passaggio precedente. Sostituisci il `targetArn` file JSON con l'ARN della distribuzione principale ed esegui il comando seguente per creare la distribuzione principale utilizzando questa nuova configurazione.

    > **Note**
    >
    > Se crei una nuova revisione di distribuzione della flotta principale, sostituisce tutte le revisioni e le distribuzioni secondarie di quella distribuzione principale. Per ulteriori informazioni, consulta Revisionare le distribuzioni.

    ```
    aws greengrassv2 create-deployment --cli-input-json file://deployment.json
    ```

    La risposta include un messaggio `deploymentId` che identifica questa distribuzione. È possibile utilizzare l'ID di distribuzione per verificare lo stato della distribuzione. Per ulteriori informazioni, consulta Controllo dello stato di implementazione.

Rivedi le distribuzioni

Ogni oggetto o gruppo di oggetti target può avere una distribuzione attiva alla volta. Quando si crea una distribuzione per una destinazione che dispone già di una distribuzione, i componenti software della nuova distribuzione sostituiscono quelli della distribuzione precedente. Se la nuova distribuzione non definisce un componente definito dalla distribuzione precedente, il software AWS IoT Greengrass Core rimuove quel componente dai dispositivi principali di destinazione. È possibile modificare una distribuzione esistente in modo da non rimuovere i componenti eseguiti sui dispositivi principali da una distribuzione precedente a una destinazione.

Per modificare una distribuzione, si crea una distribuzione che parte dagli stessi componenti e configurazioni esistenti in una distribuzione precedente. Si utilizza l'CreateDeploymentoperazione, che è la stessa operazione utilizzata per creare le distribuzioni.

Per modificare una distribuzione () AWS CLI

1. Esegui il comando seguente per elencare le distribuzioni per l'obiettivo di distribuzione. Sostituisci *targetARN* con l'ARN dell'oggetto o del gruppo di oggetti di destinazione. AWS IoT

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   La risposta contiene un elenco con la distribuzione più recente per l'obiettivo. Copia il file `deploymentId` dalla risposta da utilizzare nel passaggio successivo.

   > **Note**
   >
   > È inoltre possibile modificare una distribuzione diversa dalla revisione più recente per l'obiettivo. Specificare l'`--history-filter ALL`argomento per elencare tutte le distribuzioni per l'obiettivo. Quindi, copia l'ID della distribuzione che desideri modificare.

2. Esegui il comando seguente per ottenere i dettagli della distribuzione. Questi dettagli includono metadati, componenti e configurazione del processo. Sostituisci *DeploymentID* con l'ID del passaggio precedente.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

   La risposta contiene i dettagli della distribuzione.

3. Crea un file denominato `deployment.json` e copia la risposta del comando precedente nel file.

4. Rimuovi le seguenti coppie chiave-valore dall'oggetto JSON in `deployment.json`:

   - `deploymentId`
   - `revisionId`
   - `iotJobId`
   - `iotJobArn`
   - `creationTimestamp`
   - `isLatestForTarget`
   - `deploymentStatus`

   L'CreateDeploymentoperazione prevede un payload con la seguente struttura.

```
{
  "targetArn": "String",
  "components": Map of components,
  "deploymentPolicies": DeploymentPolicies,
  "iotJobConfiguration": DeploymentIoTJobConfiguration,
  "tags": Map of tags
}
```

5. In `deployment.json`, effettua una delle seguenti operazioni:
   - Cambia il nome della distribuzione ()`deploymentName`.
   - Modifica i componenti della distribuzione (`components`).
   - Modifica le politiche della distribuzione (`deploymentPolicies`).
   - Modifica la configurazione del lavoro della distribuzione (`iotJobConfiguration`).
   - Modifica i tag della distribuzione (`tags`).

   Per ulteriori informazioni su come definire questi dettagli di distribuzione, consultaCreare distribuzione.
6. Esegui il comando seguente per creare la distribuzione da`deployment.json`.

   ```
   aws greengrassv2 create-deployment --cli-input-json file://deployment.json
   ```

   La risposta include un messaggio `deploymentId` che identifica questa distribuzione. È possibile utilizzare l'ID di distribuzione per verificare lo stato della distribuzione. Per ulteriori informazioni, consulta Controllo dello stato di implementazione.

AnnAnnAnnAnnAnnAnnAnnAnn

È possibile annullare una distribuzione attiva per impedire l'installazione dei relativi componenti software sui dispositiviAWS IoT Greengrass principali. Se annulli una distribuzione destinata a un gruppo di oggetti, i dispositivi principali che aggiungi al gruppo non riceveranno quella distribuzione continua. Se un dispositivo principale esegue già la distribuzione, non modificherai i componenti di quel dispositivo quando annulli la distribuzione. È necessario creare una nuova distribuzione o rivederla per modificare i componenti eseguiti sui dispositivi principali che hanno ricevuto la distribuzione annullata.

Per annullare una distribuzione (AWS CLI)

1. Esegui il comando seguente per trovare l'ID dell'ultima revisione di distribuzione per una destinazione. L'ultima revisione è l'unica distribuzione che può essere attiva per un obiettivo, poiché le distribuzioni precedenti vengono annullate quando si crea una nuova revisione. *AWS IoTAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAr*

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   La risposta contiene un elenco con la distribuzione più recente per l'obiettivo. Copia il`deploymentId` dalla risposta per l'utilizzo nel passaggio successivo.

2. Usando il comando seguente per AnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnn *AnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAnnAr*

   ```
   aws greengrassv2 cancel-deployment --deployment-id deploymentId
   ```

   Se l'operazione riesce, lo stato di distribuzione cambia in`CANCELED`.

Controllo dello stato di implementazione

Puoi controllare lo stato di una implementazione in cui lo creiAWS IoT Greengrass. Puoi anche controllare lo stato deiAWS IoT processi che eseguono l'implementazione su ciascun dispositivo principale. Quando una distribuzione è attiva, lo stato delAWS IoT job è`IN_PROGRESS`. Dopo aver creato una nuova revisione di una distribuzione, lo stato delAWS IoT job della revisione precedente cambia in`CANCELLED`.

Argomenti

Controllo dello stato di implementazione

È possibile verificare lo stato di una distribuzione identificata in base alla destinazione o all'ID.

Per verificare lo stato della distribuzione per destinazione (AWS CLI)

- Eseguire il comando seguente per recuperare lo stato della implementazione più recente per un destinazioni. Sostituisci *targetArn* con l'Amazon Resource Name (ARN) dell'AWS IoToggetto o del gruppo di cose di destinazioni di implementazione.

  ```
  aws greengrassv2 list-deployments --target-arn targetArn
  ```

  La risposta contiene un elenco con l'ultima distribuzione per l'obiettivo. Questo oggetto di distribuzione include lo stato della distribuzione.

Per verificare lo stato della distribuzione in base all'ID (AWS CLI)

- Eseguire il comando seguente per recuperare lo stato di una implementazione. Sostituisci *deploymentId* con l'ID della distribuzione da interrogare.

  ```
  aws greengrassv2 get-deployment --deployment-id deploymentId
  ```

  La risposta contiene lo stato della distribuzione.

Verifica lo stato di distribuzione del dispositivo

È possibile controllare lo stato di un processo di implementazione che si applica a un singolo dispositivo di core. Puoi anche controllare lo stato di un processo di distribuzione per una distribuzione di un gruppo di oggetti.

Per verificare lo stato dei processi di distribuzione per un dispositivo principale (AWS CLI)

- Eseguire il comando seguente per recuperare lo stato di tutti i processo di implementazione per un dispositivo di core. Sostituisci *coreDeviceName*con il nome del dispositivo principale da interrogare.

  ```
  aws greengrassv2 list-effective-deployments --core-device-thing-name coreDeviceName
  ```

  La risposta contiene l'elenco dei processi di distribuzione per il dispositivo principale. È possibile identificare il lavoro da distribuire in base al processo`deploymentId` o`targetArn`. Ogni processo di implementazione contiene lo stato del processo sul dispositivo di core.

Per verificare gli stati di distribuzione di un gruppo di oggetti (AWS CLI)

1. Eseguire il comando seguente per recuperare l'ID di una implementazione esistente. Sostituisci *targetArn* con l'ARN del gruppo di destinazioni.

   ```
   aws greengrassv2 list-deployments --target-arn targetArn
   ```

   La risposta contiene un elenco con l'ultima distribuzione per l'obiettivo. Copia il`deploymentId` dalla risposta per utilizzarlo nel passaggio successivo.

   > **Note**
   >
   > Puoi anche elencare una distribuzione diversa dalla distribuzione più recente per la destinazione. Specifica l'`--history-filter ALL`argomento per elencare tutte le distribuzioni per la destinazione. Quindi, copia l'ID della distribuzione di cui desideri verificare lo stato.

2. Eseguire il comando seguente per ottenere i dettagli della implementazione. Sostituisci *DeploymentID* con l'ID della fase precedente.

   ```
   aws greengrassv2 get-deployment --deployment-id deploymentId
   ```

   La risposta contiene informazioni sulla implementazione. Copia il file`iotJobId` dalla risposta per utilizzarlo nel passaggio successivo.

3. Esegui il comando seguente per descrivere l'esecuzione del job di un dispositivo principale per la distribuzione. Sostituisci *iotJobId*e assegna un *coreDeviceThingnome* con l'ID del lavoro indicato nel passaggio precedente e il dispositivo principale di cui desideri verificare lo stato.

   ```
   aws iot describe-job-execution --job-id iotJobId --thing-name coreDeviceThingName
   ```

   La risposta contiene lo stato dell'esecuzione del processo di distribuzione del dispositivo principale e dettagli sullo stato. `detailsMap`Contiene le seguenti informazioni:

   - `detailed-deployment-status`— Lo stato di implementazione, che può avere uno dei seguenti valori:
     - `SUCCESSFUL`— La distribuzione è riuscita.

- `FAILED_NO_STATE_CHANGE`— La distribuzione non è riuscita mentre il dispositivo principale si preparava ad applicare la distribuzione.
- `FAILED_ROLLBACK_NOT_REQUESTED`— La distribuzione non è riuscita e non ha specificato di tornare a una configurazione funzionante precedente, quindi il dispositivo principale potrebbe non funzionare correttamente.
- `FAILED_ROLLBACK_COMPLETE`— L'implementazione non è riuscita e il dispositivo principale è stato ripristinato con successo a una configurazione funzionante precedente.
- `FAILED_UNABLE_TO_ROLLBACK`— La distribuzione non è riuscita e il dispositivo principale non è riuscito a ripristinare una configurazione funzionante precedente, quindi il dispositivo principale potrebbe non funzionare correttamente.

  Se la distribuzione non è riuscita, controlla il`deployment-failure-cause` valore e i file di registro del dispositivo principale per identificare il problema. Per ulteriori informazioni su come accedere ai file di registro del dispositivo di core, consultareMonitora AWS IoT Greengrass i registri.

- `deployment-failure-cause`— Un messaggio di errore che fornisce ulteriori dettagli sul motivo per cui l'esecuzione del job non è riuscita.

La risposta è simile all'esempio seguente.


```
{
  "execution": {
    "jobId": "2cc2698a-5175-48bb-adf2-1dd345606ebd",
    "status": "FAILED",
    "statusDetails": {
      "detailsMap": {
        "deployment-failure-cause": "No local or cloud component version satisfies
 the requirements. Check whether the version constraints conflict and that
 the component exists in your Account AWS with a version that matches the
 version constraints. If the version constraints conflict, revise deployments
 to resolve the conflict. Component com.example.HelloWorld version constraints:
 LOCAL_DEPLOYMENT requires =1.0.0, thinggroup/MyGreengrassCoreGroup requires
 =1.0.1.",
        "detailed-deployment-status": "FAILED_NO_STATE_CHANGE"
      }
    },
    "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
    "queuedAt": "2022-02-15T14:45:53.098000-08:00",
    "startedAt": "2022-02-15T14:46:05.670000-08:00",
```

```
    "lastUpdatedAt": "2022-02-15T14:46:20.892000-08:00",
    "executionNumber": 1,
    "versionNumber": 3
  }
}
```

Registrazione e monitoraggio in AWS IoT Greengrass

Il monitoraggio è importante per garantire l'affidabilità, la disponibilità e le prestazioni di AWS IoT Greengrass e delle soluzioni AWS. Devi raccogliere i dati sul monitoraggio da tutte le parti della soluzione AWS per eseguire più facilmente il debug di un eventuale guasto in più punti. Prima di iniziare il monitoraggio di AWS IoT Greengrass, è opportuno creare un piano di monitoraggio che includa le risposte alle seguenti domande:

- Quali sono gli obiettivi del monitoraggio?
- Quali risorse verranno monitorate?
- Con quale frequenza eseguirai il monitoraggio di queste risorse?
- Quali strumenti di monitoraggio verranno usati?
- Chi eseguirà i processi di monitoraggio?
- Chi deve ricevere una notifica quando si verifica un problema?

Argomenti

Strumenti di monitoraggio

AWS offre strumenti che puoi utilizzare per monitorare AWS IoT Greengrass. Alcuni di questi strumenti possono essere configurati per il monitoraggio automatico delle applicazioni. Alcuni degli strumenti richiedono l'intervento manuale. Si consiglia di automatizzare il più possibile i processi di monitoraggio.

È possibile utilizzare i seguenti strumenti di monitoraggio automatizzato per monitorare AWS IoT Greengrass e segnalare i problemi:

- Amazon CloudWatch Logs: monitora, archivia e accedi ai tuoi file di registro da AWS CloudTrail o altre fonti. Per ulteriori informazioni, consulta Monitoring log files nella Amazon CloudWatch User Guide.
- AWS CloudTrailMonitoraggio dei log: condividi i file di CloudTrail log tra account, monitora i file di log in tempo reale inviandoli a CloudWatch Logs, scrivi applicazioni di elaborazione dei log in Java e verifica che i file di log non siano cambiati dopo la consegna da parte di. CloudTrail Per ulteriori informazioni, consulta Lavorare con i file di CloudTrail registro nella Guida per l'AWS CloudTrailutente.
- Telemetria sanitaria del sistema Greengrass: iscriviti per ricevere i dati di telemetria inviati dal core Greengrass. Per ulteriori informazioni, consulta the section called “Raccogli dati di telemetria sanitaria del sistema”.
- Notifiche sullo stato dei dispositivi Crea eventi utilizzando Amazon EventBridge per ricevere aggiornamenti sullo stato di distribuzioni e componenti. Per ulteriori informazioni, consulta Ricevi notifiche sullo stato di implementazione e integrità dei componenti.
- Servizio Fleet Status: utilizza le operazioni dell'API Fleet Status per controllare lo stato dei dispositivi principali e dei relativi componenti Greengrass. Puoi anche visualizzare le informazioni sullo stato della flotta nella AWS IoT Greengrass console. Per ulteriori informazioni, consulta Controlla lo stato del dispositivo Greengrass core.

Monitora AWS IoT Greengrass i registri

AWS IoT Greengrass è costituito dal servizio cloud e dal software AWS IoT Greengrass Core. Il software AWS IoT Greengrass Core può scrivere log su Amazon CloudWatch Logs e sul file system locale del dispositivo principale. I componenti Greengrass eseguiti sul dispositivo principale possono anche scrivere log su CloudWatch Logs e sul file system locale. Puoi utilizzare i log di eventi per monitorare e risolvere i problemi. Tutte le voci di log di AWS IoT Greengrass includono un timestamp, un livello di log e le informazioni sull'evento.

Per impostazione predefinita, il software AWS IoT Greengrass Core scrive i log solo nel file system locale. È possibile visualizzare i log del file system in tempo reale, in modo da eseguire il debug dei componenti Greengrass sviluppati e distribuiti. Puoi anche configurare un dispositivo principale per scrivere i log in CloudWatch Logs, in modo da poter risolvere i problemi del dispositivo principale senza accedere al file system locale. Per ulteriori informazioni, consulta Abilita la registrazione nei registri CloudWatch .

Argomenti

Accedere ai log del file system

Il software AWS IoT Greengrass Core archivia i log nella */greengrass/v2*`/logs` cartella di un dispositivo principale, dove si */greengrass/v2* trova il percorso della cartella AWS IoT Greengrass principale. La cartella logs ha la seguente struttura.

```
/greengrass/v2
### logs
    ### greengrass.log
    ### greengrass_2021_09_14_15_0.log
    ### ComponentName.log
    ### ComponentName_2021_09_14_15_0.log
    ### main.log
```

- `greengrass.log`— Il file di registro del software AWS IoT Greengrass Core. Utilizzate questo file di registro per visualizzare informazioni in tempo reale su componenti e implementazioni. Questo file di registro include i registri per il nucleo Greengrass, che è il nucleo del software Core, e AWS IoT Greengrass i componenti del plug-in, come il gestore dei registri e il gestore segreto.
- *ComponentName*`.log`— File di registro dei componenti Greengrass. Utilizzate i file di registro dei componenti per visualizzare informazioni in tempo reale su un componente Greengrass in esecuzione sul dispositivo principale. I componenti generici e i componenti Lambda scrivono lo standard output (stdout) e l'errore standard (stderr) in questi file di registro.
- `main.log`— Il file di registro per il `main` servizio che gestisce i cicli di vita dei componenti. Questo file di registro sarà sempre vuoto.

Per ulteriori informazioni sulle differenze tra i componenti plug-in, generici e Lambda, consulta. Tipi di componenti

Le seguenti considerazioni si applicano quando si utilizzano i log del file system:

- Autorizzazioni utente root

  È necessario disporre delle autorizzazioni root, per poter leggere i log di AWS IoT Greengrass sul file system.

- Rotazione dei file di registro

  Il software AWS IoT Greengrass Core ruota i file di registro ogni ora o quando superano un limite di dimensione del file. I file di registro ruotati contengono un timestamp nel nome del file. Ad esempio, potrebbe essere denominato un file di registro AWS IoT Greengrass del software Core ruotato. `greengrass_2021_09_14_15_0.log` Il limite di dimensione del file predefinito è 1.024 KB (1 MB). È possibile configurare il limite di dimensione del file sul componente Greengrass nucleus.

- Eliminazione dei file di registro

  Il software AWS IoT Greengrass Core pulisce i file di registro precedenti quando le dimensioni dei file di registro del software AWS IoT Greengrass Core o dei file di registro dei componenti Greengrass, inclusi i file di registro ruotati, superano il limite di spazio su disco. Il limite di spazio su disco predefinito per il registro del software AWS IoT Greengrass Core e per ogni registro dei componenti è di 10.240 KB (10 MB). È possibile configurare il limite di spazio su disco del registro del software AWS IoT Greengrass Core sul componente Greengrass nucleus o sul componente log manager. È possibile configurare il limite di spazio su disco di registro di ciascun componente sul componente log manager.

Per visualizzare il file di registro del software AWS IoT Greengrass Core

- Eseguite il comando seguente per visualizzare il file di registro in tempo reale. Sostituisci */greengrass/v2* con il percorso della cartella AWS IoT Greengrass principale.

  Linux or Unix

  ```
  sudo tail -f /greengrass/v2/logs/greengrass.log
  ```

  Windows Command Prompt (CMD)

  ```
  type C:\greengrass\v2\logs\com.example.HelloWorld.log
  ```

  Il `type` comando scrive il contenuto del file nel terminale. Esegui questo comando più volte per osservare le modifiche nel file.

PowerShell

```
gc C:\greengrass\v2\logs\greengrass.log -Tail 10 -Wait
```

Per visualizzare il file di registro di un componente

- Esegui il comando seguente per visualizzare il file di registro in tempo reale. Sostituisci */greengrass/v2* o *C:\greengrass\v2* con il percorso della cartella AWS IoT Greengrass principale e sostituisci *com.example. HelloWorld*con il nome del componente.

  Linux or Unix

  ```
  sudo tail -f /greengrass/v2/logs/com.example.HelloWorld.log
  ```

  PowerShell

  ```
  gc C:\greengrass\v2\logs\com.example.HelloWorld.log -Tail 10 -Wait
  ```

Puoi anche usare il `logs` comando della Greengrass CLI per analizzare i log di Greengrass su un dispositivo principale. Per utilizzare il `logs` comando, è necessario configurare il nucleo Greengrass per l'output di file di registro in formato JSON. Per ulteriori informazioni, consultare Interfaccia a riga di comando Greengrass e log.

Registri di accesso CloudWatch

È possibile implementare il componente di gestione dei registri per configurare il dispositivo principale per la scrittura nei registri. CloudWatch Per ulteriori informazioni, consulta Abilita la registrazione nei registri CloudWatch . Quindi, puoi visualizzare i log nella pagina Logs della CloudWatch console Amazon o utilizzando l'API CloudWatch Logs.

Nome del gruppo di log

```
/aws/greengrass/componentType/region/componentName
```

Il nome del gruppo di log utilizza le seguenti variabili:

- `componentType`— Il tipo di componente, che può essere uno dei seguenti:

  - `GreengrassSystemComponent`— Questo gruppo di log include i log per i componenti del nucleo e del plugin, che vengono eseguiti nella stessa JVM del nucleo Greengrass. Il componente fa parte del nucleo Greengrass.
  - `UserComponent`— Questo gruppo di log include i registri per componenti generici, componenti Lambda e altre applicazioni sul dispositivo. Il componente non fa parte del nucleo Greengrass.

  Per ulteriori informazioni, consulta Tipi di componenti.
- `region`— La AWS regione utilizzata dal dispositivo principale.
- `componentName`— Il nome del componente. Per i registri di sistema, questo valore è`System`.

Nome del flusso di log

```
/date/thing/thingName
```

Il nome del flusso di registro utilizza le seguenti variabili:

- `date`— La data del registro, ad esempio`2020/12/15`. Il componente log manager utilizza il `yyyy/MM/dd` formato.
- `thingName`— Il nome del dispositivo principale.

> **Note**
>
> Se il nome di un oggetto contiene due punti (`:`), il gestore dei registri sostituisce i due punti con un segno più (+).

Le seguenti considerazioni si applicano quando si utilizza il componente log manager per scrivere CloudWatch su Logs:

- Registra i ritardi

> **Note**
>
> Si consiglia di eseguire l'aggiornamento alla versione 2.3.0 di log manager, che riduce i ritardi di registro per i file di registro ruotati e attivi. Quando esegui l'aggiornamento a log manager 2.3.0, ti consigliamo di eseguire anche l'aggiornamento a Greengrass nucleus 2.9.1.

Il componente di log manager versione 2.2.8 (e precedenti) elabora e carica i log solo dai file di registro ruotati. Per impostazione predefinita, il software AWS IoT Greengrass Core ruota i file di registro ogni ora o dopo che hanno raggiunto i 1.024 KB. Di conseguenza, il componente log manager carica i log solo dopo che il software AWS IoT Greengrass Core o un componente Greengrass hanno scritto oltre 1.024 KB di log. È possibile configurare un limite inferiore per le dimensioni dei file di registro per far sì che i file di registro ruotino più spesso. Ciò fa sì che il componente log manager carichi i log in Logs più CloudWatch frequentemente.

Il componente di gestione dei registri versione 2.3.0 (e successive) elabora e carica tutti i log. Quando si scrive un nuovo registro, la versione 2.3.0 (e successive) di log manager elabora e carica direttamente il file di registro attivo anziché attendere che venga ruotato. Ciò significa che è possibile visualizzare il nuovo registro in 5 minuti o meno.

Il componente di gestione dei registri carica periodicamente nuovi registri. Per impostazione predefinita, il componente di gestione dei registri carica nuovi registri ogni 5 minuti. È possibile configurare un intervallo di caricamento inferiore, in modo che il componente log manager carichi i log in Logs più frequentemente configurando CloudWatch . `periodicUploadIntervalSec` Per ulteriori informazioni su come configurare questo intervallo periodico, vedere Configurazione.

I log possono essere caricati quasi in tempo reale dallo stesso file system Greengrass. Se hai bisogno di osservare i log in tempo reale, prendi in considerazione l'utilizzo dei log del file system.

> **Note**
>
> Se utilizzi file system diversi su cui scrivere i log, log manager torna al comportamento delle versioni 2.2.8 e precedenti dei componenti di log manager. Per informazioni sull'accesso ai log del file system, consulta Access file system logs.

- Inclinazione dell'orologio

  Il componente log manager utilizza il processo di firma standard Signature Version 4 per creare richieste API per CloudWatch Logs. Se l'ora del sistema su un dispositivo principale non è sincronizzata per più di 15 minuti, CloudWatch Logs rifiuta le richieste. Per ulteriori informazioni, consulta la pagina relativa al processo di firma Signature Version 4 nella Riferimenti generali di AWS.

Accedere ai registri dei servizi di sistema

Se si configura il software AWS IoT Greengrass Core come servizio di sistema, è possibile visualizzare i registri dei servizi di sistema per risolvere problemi, ad esempio il mancato avvio del software.

Per visualizzare i registri dei servizi di sistema (CLI)

1. Esegui il comando seguente per visualizzare i registri dei servizi del sistema software AWS IoT Greengrass Core.

   Linux or Unix (systemd)

   ```
   sudo journalctl -u greengrass.service
   ```

   Windows Command Prompt (CMD)

   ```
   type C:\greengrass\v2\logs\greengrass.wrapper.log
   ```

   PowerShell

   ```
   gc C:\greengrass\v2\logs\greengrass.wrapper.log
   ```

2. Sui dispositivi Windows, il software AWS IoT Greengrass Core crea un file di registro separato per gli errori del servizio di sistema. Eseguite il comando seguente per visualizzare i log degli errori del servizio di sistema.

   Windows Command Prompt (CMD)

   ```
   type C:\greengrass\v2\logs\greengrass.err.log
   ```

   PowerShell

   ```
   gc C:\greengrass\v2\logs\greengrass.err.log
   ```

Sui dispositivi Windows, è inoltre possibile utilizzare l'applicazione Event Viewer per visualizzare i registri dei servizi di sistema.

Per visualizzare i registri dei servizi di Windows (Event Viewer)

1. Aprire l'applicazione Event Viewer.
2. Seleziona Windows Logs per espanderlo.
3. Scegli Applicazione per visualizzare i registri dei servizi dell'applicazione.
4. Trova e apri i registri degli eventi la cui origine è. greengrass

Abilita la registrazione nei registri CloudWatch

È possibile implementare il componente di gestione dei registri per configurare un dispositivo principale per scrivere i log nei registri. CloudWatch È possibile abilitare CloudWatch i registri del software Logs for AWS IoT Greengrass Core e abilitare i CloudWatch log per componenti Greengrass specifici.

> **Note**
>
> Il ruolo di scambio di token del dispositivo principale Greengrass deve consentire al dispositivo principale di scrivere su CloudWatch Logs, come mostrato nell'esempio seguente di politica IAM. Se hai installato il software AWS IoT Greengrass Core con il provisioning automatico delle risorse, il tuo dispositivo principale dispone di queste autorizzazioni.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:logs:*:*:*"
    }
  ]
}
```

Per configurare un dispositivo principale in modo che scriva i log del software AWS IoT Greengrass Core in CloudWatch Logs, crea una distribuzione che specifichi un aggiornamento della configurazione impostato `uploadToCloudWatch` su per il componente. `true` `aws.greengrass.LogManager` AWS IoT Greengrass I registri del software di base includono i registri per il nucleo Greengrass e i componenti del plug-in.

```
{
  "logsUploaderConfiguration": {
    "systemLogsConfiguration": {
      "uploadToCloudWatch": "true"
    }
  }
}
```

Per configurare un dispositivo principale per scrivere i log di un componente Greengrass in CloudWatch Logs, crea una distribuzione che specifichi un aggiornamento della configurazione che aggiunga il componente all'elenco delle configurazioni di registrazione dei componenti. Quando aggiungete un componente a questo elenco, il componente di gestione dei log scrive i registri in Logs. CloudWatch I registri dei componenti includono i registri dei componenti generici e dei componenti Lambda.

```
{
  "logsUploaderConfiguration": {
    "componentLogsConfigurationMap": {
      "com.example.HelloWorld": {

      }
    }
  }
}
```

Quando si distribuisce il componente Log Manager, è anche possibile configurare i limiti di spazio su disco e stabilire se il dispositivo principale elimina i file di registro dopo averli scritti in Logs. CloudWatch Per ulteriori informazioni, consulta Configurazione della registrazione per AWS IoT Greengrass.

Configurazione della registrazione per AWS IoT Greengrass

È possibile configurare le seguenti opzioni per personalizzare la registrazione per i dispositivi core Greengrass. Per configurare queste opzioni, crea una distribuzione che specifichi un aggiornamento della configurazione per i componenti Greengrass nucleus o log manager.

- Scrivere i log in Logs CloudWatch

  Per risolvere in remoto i problemi dei dispositivi principali, è possibile configurare i dispositivi principali per scrivere il software AWS IoT Greengrass Core e i registri dei componenti in Logs. CloudWatch A tale scopo, distribuisci e configura il componente log manager. Per ulteriori informazioni, consulta Abilita la registrazione nei registri CloudWatch .

- Eliminazione dei file di registro caricati

  Per ridurre l'utilizzo dello spazio su disco, è possibile configurare i dispositivi principali per eliminare i file di registro dopo averli scritti in CloudWatch Logs. Per ulteriori informazioni, consultate il `deleteLogFileAfterCloudUpload` parametro del componente Log Manager, che potete specificare per i log del software AWS IoT Greengrass Core e i log dei componenti.

- Registra i limiti di spazio su disco

  Per limitare l'utilizzo dello spazio su disco, è possibile configurare lo spazio massimo su disco per ogni registro, compresi i file di registro ruotati, su un dispositivo principale. Ad esempio, è possibile configurare lo spazio massimo combinato su disco `greengrass.log` e i file ruotati. `greengrass.log` Per ulteriori informazioni, consultate il parametro del componente Greengrass nucleus e il `logging.totalLogsSizeKB` parametro del componente log manager, che potete specificare per i log del `diskSpaceLimit`software AWS IoT Greengrass Core e i log dei componenti.

- Limite di dimensione del file di registro

  È possibile configurare la dimensione massima del file per ogni file di registro. Dopo che un file di registro supera questo limite di dimensioni, il software AWS IoT Greengrass Core crea un nuovo file di registro. Il componente di gestione dei registri versione 2.28 (e precedenti) scrive solo file di registro ruotati CloudWatch nei registri, quindi è possibile specificare un limite di dimensione inferiore per scrivere i log nei registri con maggiore frequenza. CloudWatch La versione 2.3.0 (e successive) del componente di gestione dei log elabora e carica tutti i log invece di attendere che vengano ruotati. Per ulteriori informazioni, vedete il parametro limite di dimensione del file di registro del componente Greengrass nucleus (). `logging.fileSizeKB`

- Livelli minimi di registro

  È possibile configurare il livello minimo di log che il componente Greengrass nucleus scrive nei log del file system. Ad esempio, è possibile specificare registri di `DEBUG` livello per facilitare la risoluzione dei problemi oppure specificare registri di `ERROR` livello per ridurre la quantità di log creati da un dispositivo principale. Per ulteriori informazioni, vedete il parametro log level del componente Greengrass nucleus (). `logging.level`

  È inoltre possibile configurare il livello minimo di registro che il componente log manager scrive in Logs. CloudWatch Ad esempio, è possibile specificare un livello di registro più elevato per ridurre i costi di registrazione. Per ulteriori informazioni, consultate il `minimumLogLevel` parametro del componente Log Manager, che potete specificare per i log del software AWS IoT Greengrass Core e i log dei componenti.

- Intervallo per verificare la presenza di log da scrivere nei registri CloudWatch

  Per aumentare o diminuire la frequenza con cui il componente di gestione dei log scrive i log nei CloudWatch registri, è possibile configurare l'intervallo in cui verifica la presenza di nuovi file di registro da scrivere. Ad esempio, è possibile specificare un intervallo inferiore per visualizzare i log nei CloudWatch registri prima di quanto si farebbe con l'intervallo predefinito di 5 minuti. È possibile specificare un intervallo più elevato per ridurre i costi, poiché il componente di gestione dei registri raggruppa i file di registro in un numero inferiore di richieste. Per ulteriori informazioni, vedete il parametro dell'intervallo di caricamento del componente log manager (). `periodicUploadIntervalSec`

- Formato del registro

  È possibile scegliere se il software AWS IoT Greengrass Core scrive i log in formato testo o JSON. Scegliete il formato di testo se leggete i log o scegliete il formato JSON se utilizzate un'applicazione per leggere o analizzare i log. Per ulteriori informazioni, vedete il parametro di formato log del componente Greengrass nucleus (). `logging.format`

- Cartella dei registri del file system locale

  È possibile modificare la cartella dei registri */greengrass/v2*`/logs` da un'altra cartella sul dispositivo principale. Per ulteriori informazioni, vedete il parametro della directory di output del componente Greengrass nucleus (). `logging.outputDirectory`

Log AWS CloudTrail

AWS IoT Greengrasssi integra conAWS CloudTrail, un servizio che fornisce una registrazione delle azioni intraprese da un utente, da un ruolo o da un partecipante. Servizio AWS AWS IoT Greengrass Per ulteriori informazioni, consulta Registra le chiamate AWS IoT Greengrass V2 API con AWS CloudTrail.

Registra le chiamate AWS IoT Greengrass V2 API con AWS CloudTrail

AWS IoT Greengrass V2 è integrato con AWS CloudTrail, un servizio che fornisce una registrazione delle azioni intraprese da un utente, ruolo o AWS servizio in AWS IoT Greengrass Version 2. CloudTrail acquisisce tutte le chiamate API AWS IoT Greengrass come eventi. Le chiamate acquisite includono chiamate dalla AWS IoT Greengrass console e chiamate di codice alle operazioni AWS IoT Greengrass API.

Se crei un trail, puoi abilitare la consegna continua di CloudTrail eventi a un bucket S3, inclusi gli eventi per. AWS IoT Greengrass Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, puoi determinare a quale richiesta è stata inviata AWS IoT Greengrass, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi.

Per ulteriori informazioni in merito CloudTrail, consulta la Guida AWS CloudTrail per l'utente.

AWS IoT Greengrass V2 informazioni in CloudTrail

CloudTrail è abilitato sul tuo account al Account AWS momento della creazione dell'account. Quando si verifica un'attività in AWS IoT Greengrass, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi AWS di servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare gli eventi recenti in Account AWS. Per ulteriori informazioni, consulta Visualizzazione degli eventi con la cronologia degli CloudTrail eventi.

Per una registrazione continua degli eventi del tuo sito Account AWS, inclusi gli eventi di AWS IoT Greengrass, crea un percorso. Un trail consente di CloudTrail inviare file di registro a un bucket S3. Per impostazione predefinita, quando crei un trail nella console, il trail si applica a tutti i Regione AWS file. Il trail registra gli eventi da tutte le regioni della AWS partizione e consegna i file di registro al

bucket S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta gli argomenti seguenti:

- Panoramica della creazione di un percorso
- CloudTrail servizi e integrazioni supportati
- Configurazione delle notifiche Amazon SNS per CloudTrail
- Ricezione di file di CloudTrail registro da più regioni e ricezione di file di CloudTrail registro da più account

Tutte AWS IoT Greengrass V2 le azioni vengono registrate CloudTrail e documentate nell'AWS IoT Greengrass V2 API Reference. Ad esempio, le chiamate a `CreateDeployment` e `CreateComponentVersion` le `CancelDeployment` azioni generano voci nei file di CloudTrail registro.

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente root o AWS Identity and Access Management (IAM).
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro AWS servizio.

Per ulteriori informazioni, consulta Elemento CloudTrail userIdentity.

AWS IoT Greengrass eventi di dati in CloudTrail

Gli eventi relativi ai dati forniscono informazioni sulle operazioni eseguite sulle risorse su o all'interno di una risorsa (ad esempio, l'ottenimento della versione di un componente o la configurazione di una distribuzione). Queste operazioni sono definite anche operazioni del piano dei dati. Gli eventi di dati sono spesso attività che interessano volumi elevati di dati. Per impostazione predefinita, CloudTrail non registra gli eventi relativi ai dati. La cronologia CloudTrail degli eventi non registra gli eventi relativi ai dati.

Per gli eventi di dati sono previsti costi aggiuntivi. Per ulteriori informazioni sui CloudTrail prezzi, consulta la sezione AWS CloudTrail Prezzi.

Puoi registrare gli eventi relativi ai dati per i tipi di AWS IoT Greengrass risorse utilizzando la CloudTrail console o AWS CLI le operazioni CloudTrail dell'API. La tabella in questa sezione mostra i tipi di risorse disponibili per AWS IoT Greengrass.

- Per registrare gli eventi relativi ai dati utilizzando la CloudTrail console, crea un percorso o un data store di eventi per registrare gli eventi di dati oppure aggiorna un trail o un data store di eventi esistente per registrare gli eventi di dati.

  1. Scegli Data events per registrare gli eventi relativi ai dati.
  2. Dall'elenco Tipo di evento Data, scegli il tipo di risorsa per il quale desideri registrare gli eventi relativi ai dati.
  3. Scegli il modello di selettore di registro che desideri utilizzare. Puoi registrare tutti gli eventi relativi ai dati per il tipo di risorsa, registrare tutti `readOnly` gli eventi, registrare tutti `writeOnly` gli eventi o creare un modello di selettore di registro personalizzato per filtrare i `readOnly` campi`eventName`, `eresources.ARN`.

- Per registrare gli eventi relativi ai dati utilizzando il AWS CLI, configura il `--advanced-event-selectors` parametro in modo che il `eventCategory` campo sia uguale `Data` e il `resources.type` campo uguale al valore del tipo di risorsa (vedi tabella). È possibile aggiungere condizioni per filtrare i valori dei `resources.ARN` campi `readOnly`e`eventName`, e.
  - Per configurare un percorso per registrare gli eventi relativi ai dati, esegui il put-event-selectorscomando. Per ulteriori informazioni, vedere Registrazione degli eventi relativi ai dati per i AWS CLI sentieri con.
  - Per configurare un Event Data Store per registrare gli eventi di dati, esegui il create-event-data-storecomando per creare un nuovo Event Data Store per registrare gli eventi di dati oppure esegui il update-event-data-storecomando per aggiornare un Event Data Store esistente. Per ulteriori informazioni, vedere Registrazione degli eventi di dati per i data store di eventi con. AWS CLI

La tabella seguente elenca i tipi di AWS IoT Greengrass risorse. La colonna Data event type (console) mostra il valore da scegliere dall'elenco Data event type sulla CloudTrail console. La colonna del valore resources.type mostra il `resources.type` valore da specificare durante la configurazione dei selettori di eventi avanzati utilizzando le API o. AWS CLI CloudTrail La CloudTrail colonna Data API loggate mostra le chiamate API registrate per il tipo di risorsa. CloudTrail

Tipo di evento di dati (console)	valore resources.type	API di dati registrate su CloudTrail
Certificato IoT	`AWS::IoT::Certificate`	• VerifyClientDeviceIdentity • VerifyClientDeviceIoTCertificateAssociation
Versione del componente IoT Greengrass	`AWS::GreengrassV2::ComponentVersion`	• ResolveComponentCandidates
Implementazione di IoT Greengrass	`AWS::GreengrassV2::Deployment`	• GetDeploymentConfiguration
Cosa IoT	`AWS::IoT::Thing`	• ListThingGroupsForCoreDevices • PutCertificateAuthorities • VerifyClientDeviceIoTCertificateAssociation

> **Note**
>
> Greengrass non registra gli eventi di accesso negato.

Puoi configurare selettori di eventi avanzati per filtrare `resources.ARN` i campi `eventNamereadOnly`, e per registrare solo gli eventi che ritieni importanti.

Aggiungi un filtro `eventName` per includere o escludere API di dati specifiche.

Per ulteriori informazioni sui campi, consulta AdvancedFieldSelector.

Gli esempi seguenti mostrano come configurare i selettori avanzati utilizzando. AWS CLI*`TrailName`*Sostituisci una *`regione`* con le tue informazioni.

Example — Registra gli eventi relativi ai dati per gli oggetti IoT

```
aws cloudtrail put-event-selectors --trail-name TrailName --region region \
--advanced-event-selectors \
```

```
‘[
    {
            “Name”: “Log all thing data events”,
            “FieldSelectors”: [
                { “Field”: “eventCategory”, “Equals”: [“Data”] },
                { “Field”: “resources.type”, “Equals”: [“AWS::IoT::Thing”] }
            ]
        }
]’
```

Example — Filtra in base a una specifica API IoT

```
aws cloudtrail put-event-selectors --trail-name TrailName --region region \
--advanced-event-selectors \
‘[
    {
            “Name”: “Log IoT Greengrass PutCertificateAuthorities API calls”,
            “FieldSelectors”: [
                { “Field”: “eventCategory”, “Equals”: [“Data”] },
                { “Field”: “resources.type”, “Equals”: [“AWS::IoT::Thing”] },
                { “Field”: “eventName”, “Equals”: [“PutCertificateAuthorities”] }
            ]
    }
]’
```

Example — Registra tutti gli eventi relativi ai dati Greengrass

```
aws cloudtrail put-event-selectors --trail-name TrailName --region region \
--advanced-event-selectors \
‘[
    {
        “Name”: “Log all certificate data events”,
        “FieldSelectors”: [
            {
                “Field”: “eventCategory”,
                “Equals”: [
                    “Data”
                ]
            },
            {
                “Field”: “resources.type”,
                “Equals”: [
                    “AWS::IoT::Certificate”
```

```
                ]
            }
        ]
    },
    {
        “Name”: “Log all component version data events”,
        “FieldSelectors”: [
            {
                “Field”: “eventCategory”,
                “Equals”: [
                    “Data”
                ]
            },
            {
                “Field”: “resources.type”,
                “Equals”: [
                    “AWS::GreengrassV2::ComponentVersion”
                ]
            }
        ]
    },
    {
        “Name”: “Log all deployment version”,
        “FieldSelectors”: [
            {
                “Field”: “eventCategory”,
                “Equals”: [
                    “Data”
                ]
            },
            {
                “Field”: “resources.type”,
                “Equals”: [
                    “AWS::GreengrassV2::Deployment”
                ]
            }
        ]
    },
    {
        “Name”: “Log all thing data events”,
        “FieldSelectors”: [
            {
                “Field”: “eventCategory”,
                “Equals”: [
```

```
                    “Data”
                ]
            },
            {
                “Field”: “resources.type”,
                “Equals”: [
                    “AWS::IoT::Thing”
                ]
            }
        ]
    }
]’
```

AWS IoT Greengrass eventi di gestione in CloudTrail

Gli eventi di gestione forniscono informazioni sulle operazioni di gestione eseguite sulle risorse AWS dell'account. Queste operazioni sono definite anche operazioni del piano di controllo (control-plane). Per impostazione predefinita, CloudTrail registra gli eventi di gestione.

AWS IoT Greengrass registra tutte le operazioni AWS IoT Greengrass del piano di controllo come eventi di gestione. Per un elenco delle operazioni del piano di AWS IoT Greengrass controllo a cui si AWS IoT Greengrass effettua l'accesso CloudTrail, vedere il riferimento all'AWS IoT Greengrass API, versione 2.

Comprendere le AWS IoT Greengrass V2 voci dei file di registro

Un trail è una configurazione che consente la consegna di eventi come file di registro a un bucket S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta da un'origine. Include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che illustra l'`CreateDeployment`azione.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/Administrator",
```

```
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "Administrator"
    },
    "eventTime": "2021-01-06T02:38:05Z",
    "eventSource": "greengrass.amazonaws.com",
    "eventName": "CreateDeployment",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "203.0.113.0",
    "userAgent": "aws-cli/2.1.9 Python/3.7.9 Windows/10 exe/AMD64 prompt/off command/
greengrassv2.create-deployment",
    "requestParameters": {
        "deploymentPolicies": {
            "failureHandlingPolicy": "DO_NOTHING",
            "componentUpdatePolicy": {
                "timeoutInSeconds": 60,
                "action": "NOTIFY_COMPONENTS"
            },
            "configurationValidationPolicy": {
                "timeoutInSeconds": 60
            }
        },
        "deploymentName": "Deployment for MyGreengrassCoreGroup",
        "components": {
            "aws.greengrass.Cli": {
                "componentVersion": "2.0.3"
            }
        },
        "iotJobConfiguration": {},
        "targetArn": "arn:aws:iot:us-west-2:123456789012:thinggroup/
MyGreengrassCoreGroup"
    },
    "responseElements": {
        "iotJobArn": "arn:aws:iot:us-west-2:123456789012:job/fdfeba1d-ac6d-44ef-
ab28-54f684ea578d",
        "iotJobId": "fdfeba1d-ac6d-44ef-ab28-54f684ea578d",
        "deploymentId": "4196dddc-0a21-4c54-a985-66a525f6946e"
    },
    "requestID": "311b9529-4aad-42ac-8408-c06c6fec79a9",
    "eventID": "c0f3aa2c-af22-48c1-8161-bad4a2ab1841",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
```

```
    "recipientAccountId": "123456789012"
}
```

Raccogli dati di telemetria sanitaria del sistema dai dispositivi principali AWS IoT Greengrass

I dati di telemetria sullo stato del sistema sono dati diagnostici che possono aiutarti a monitorare le prestazioni delle operazioni critiche sui tuoi dispositivi principali Greengrass. È possibile creare progetti e applicazioni per recuperare, analizzare, trasformare e generare report sui dati di telemetria dai dispositivi periferici. Gli esperti del settore, come gli ingegneri di processo, possono utilizzare queste applicazioni per ottenere informazioni sullo stato della flotta.

È possibile utilizzare i seguenti metodi per raccogliere dati di telemetria dai dispositivi principali Greengrass:

- Componente emettitore di telemetria Nucleus: il componente emettitore di telemetria nucleus (`aws.greengrass.telemetry.NucleusEmitter`) su un dispositivo core Greengrass pubblica i dati di telemetria sull'argomento per impostazione predefinita. `$local/greengrass/telemetry` Puoi utilizzare i dati pubblicati su questo argomento per agire localmente sul tuo dispositivo principale, anche quando la connettività del dispositivo al cloud è limitata. Facoltativamente, puoi anche configurare il componente per pubblicare dati di telemetria su un argomento AWS IoT Core MQTT di tua scelta.

  È necessario distribuire il componente Nucleus Emitter su un dispositivo principale per pubblicare i dati di telemetria. Non sono previsti costi associati alla pubblicazione dei dati di telemetria sull'argomento locale. Tuttavia, l'uso di un argomento MQTT per la pubblicazione di dati su Cloud AWS è soggetto ai prezzi. AWS IoT Core

  AWS IoT Greengrassfornisce diversi componenti della community per aiutarti ad analizzare e visualizzare i dati di telemetria localmente sul tuo dispositivo principale utilizzando InfluxDB e Grafana. Questi componenti utilizzano i dati di telemetria del componente emettitore del nucleo. Per ulteriori informazioni, consultate il README per il componente publisher InfluxDB.
- Agente di telemetria: l'agente di telemetria sui dispositivi core Greengrass raccoglie i dati di telemetria locali e li pubblica su Amazon senza richiedere alcuna interazione con il cliente. EventBridge I dispositivi principali pubblicano i dati EventBridge di telemetria con la massima diligenza possibile. Ad esempio, i dispositivi principali potrebbero non riuscire a fornire dati di telemetria mentre sono offline.

La funzionalità dell'agente di telemetria è abilitata per impostazione predefinita per tutti i dispositivi core Greengrass. Si inizia automaticamente a ricevere dati non appena si configura un dispositivo Greengrass core. Oltre ai costi del collegamento dati, il trasferimento dei dati dal dispositivo principale a AWS IoT Core è gratuito. Questo perché l'agente pubblica su un argomento AWS riservato. Tuttavia, a seconda del caso d'uso, potrebbero verificarsi dei costi quando si ricevono o si elaborano i dati.

> **Note**
>
> Amazon EventBridge è un servizio di bus per eventi che puoi utilizzare per connettere le tue applicazioni con dati provenienti da diverse fonti, come i dispositivi core Greengrass. Per ulteriori informazioni, consulta What is Amazon EventBridge? nella Amazon EventBridge User Guide.

Per garantire il corretto funzionamento del software AWS IoT Greengrass Core, AWS IoT Greengrass utilizza i dati per scopi di sviluppo e miglioramento della qualità. Questa funzionalità aiuta anche a sviluppare nuove e migliorate funzionalità edge. AWS IoT Greengrassconserva i dati di telemetria per un massimo di sette giorni.

Questa sezione descrive come configurare e utilizzare l'agente di telemetria. Per informazioni sulla configurazione del componente dell'emettitore di telemetria Nucleus, vedere. Emettitore di telemetria Nucleus

Argomenti

Metriche di telemetria

La tabella seguente descrive le metriche pubblicate dall'agente di telemetria.

Nome	Descrizione	
System (Sistema)		

Nome	Descrizione
`SystemMemUsage`	La quantità di memoria attualmente utilizzata da tutte le applicazioni sul dispositivo principale Greengrass, incluso il sistema operativo.
`CpuUsage`	La quantità di CPU attualmente utilizzata da tutte le applicazioni sul dispositivo principale Greengrass, incluso il sistema operativo.
`TotalNumberOfFDs`	Il numero di descrittori di file memorizzati dal sistema operativo del dispositivo principale Greengrass. Un descrittore di file identifica in modo univoco un file aperto.
Nucleo Greengrass	
`NumberOfComponentsRunning`	Il numero di componenti in esecuzione sul dispositivo principale Greengrass.
`NumberOfComponentsErrored`	Il numero di componenti in stato di errore sul dispositivo principale Greengrass.
`NumberOfComponentsInstalled`	Il numero di componenti installati sul dispositivo principale Greengrass.
`NumberOfComponentsStarting`	Il numero di componenti che si avviano sul dispositivo principale Greengrass.

Nome	Descrizione	
`NumberOfComponents New`	Il numero di componenti nuovi sul dispositivo principale Greengrass.	
`NumberOfComponents Stopping`	Il numero di componenti che si arrestano sul dispositivo principale Greengrass.	
`NumberOfComponents Finished`	Il numero di componenti completati sul dispositivo principale Greengrass.	
`NumberOfComponents Broken`	Il numero di componenti danneggiati sul dispositivo principale Greengrass.	
`NumberOfComponents Stateless`	Il numero di component i stateless sul dispositivo principale Greengrass.	
Autenticazione del dispositi vo client: questa funzionalità richiede la versione 2.4.0 o successiva del componente di autenticazione del dispositivo client.		
`VerifyClientDevice Identity.Success`	Il numero di volte in cui la verifica dell'identità del dispositivo client ha avuto esito positivo.	
`VerifyClientDevice Identity.Failure`	Il numero di volte in cui la verifica dell'identità del dispositivo client non è riuscita.	

Nome	Descrizione
`AuthorizeClientDeviceActions.Success`	Il numero di volte in cui il dispositivo client è autorizzato a completare le azioni richieste.
`AuthorizeClientDeviceActions.Failure`	Il numero di volte in cui il dispositivo client non è autorizzato a completare le azioni richieste.
`GetClientDeviceAuthToken.Success`	Il numero di volte in cui il dispositivo client viene autenticato con successo.
`GetClientDeviceAuthToken.Failure`	Il numero di volte in cui il dispositivo client non può essere autenticato.
`SubscribeToCertificateUpdates.Success`	Il numero di sottoscrizioni riuscite agli aggiornamenti dei certificati.
`SubscribeToCertificateUpdates.Failure`	Il numero di tentativi falliti di sottoscrizione agli aggiornamenti dei certificati.
`ServiceError`	Il numero di errori interni non gestiti nell'autenticazione del dispositivo client.
Stream manager: questa funzionalità richiede la versione 2.7.0 o successiva del componente Greengrass nucleus.	

Nome	Descrizione
`BytesAppended`	Il numero di byte di dati aggiunti allo stream manager.
`BytesUploadedToIoTAnalytics`	Il numero di byte di dati che Stream Manager esporta nei canali in cui vengono esportati . AWS IoT Analytics
`BytesUploadedToKinesis`	Il numero di byte di dati che Stream Manager esporta in flussi in Amazon Kinesis Data Streams.
`BytesUploadedToIoTSiteWise`	Il numero di byte di dati in cui Stream Manager esporta nelle proprietà degli asset. AWS IoT SiteWise
`BytesUploadedToS3`	Il numero di byte di dati che Stream Manager esporta in oggetti in Amazon S3.

Configura le impostazioni dell'agente di telemetria

L'agente di telemetria utilizza le seguenti impostazioni predefinite:

- L'agente di telemetria aggrega i dati di telemetria ogni ora.
- L'agente di telemetria pubblica un messaggio di telemetria ogni 24 ore.

L'agente di telemetria pubblica i dati utilizzando il protocollo MQTT con un livello di qualità del servizio (QoS) pari a 0, il che significa che non conferma la consegna né riprova i tentativi di pubblicazione. I messaggi di telemetria condividono una connessione MQTT con altri messaggi per gli abbonamenti a cui sono destinati. AWS IoT Core

Oltre ai costi del collegamento dati, il trasferimento dei dati dal core all'altro è gratuito. AWS IoT Core Questo perché l'agente pubblica su un argomento AWS riservato. Tuttavia, a seconda del caso d'uso, potrebbero verificarsi dei costi quando si ricevono o si elaborano i dati.

È possibile abilitare o disabilitare la funzionalità dell'agente di telemetria per ogni dispositivo principale Greengrass. È inoltre possibile configurare gli intervalli in base ai quali il dispositivo principale aggrega e pubblica i dati. Per configurare la telemetria, personalizza il parametro di configurazione della telemetria quando distribuisci il componente Greengrass nucleus.

Iscriviti ai dati di telemetria in EventBridge

Puoi creare regole in Amazon EventBridge che definiscono come elaborare i dati di telemetria pubblicati dall'agente di telemetria sul dispositivo principale Greengrass. Quando EventBridge riceve i dati, richiama le azioni mirate definite nelle regole. Ad esempio, è possibile creare regole relative agli eventi che inviano notifiche, archiviano informazioni sugli eventi, intraprendono azioni correttive o richiamano altri eventi.

Eventi di telemetria

Gli eventi di telemetria utilizzano il formato seguente.

```
{
  "version": "0",
  "id": "a09d303e-2f6e-3d3c-a693-8e33f4fe3955",
  "detail-type": "Greengrass Telemetry Data",
  "source": "aws.greengrass",
  "account": "123456789012",
  "time": "2020-11-30T20:45:53Z",
  "region": "us-east-1",
  "resources": [],
  "detail": {
    "ThingName": "MyGreengrassCore",
    "Schema": "2020-07-30",
    "ADP": [
      {
        "TS": 1602186483234,
        "NS": "SystemMetrics",
        "M": [
          {
            "N": "TotalNumberOfFDs",
            "Sum": 6447.0,
            "U": "Count"
```

```
          },
          {
            "N": "CpuUsage",
            "Sum": 15.458333333333332,
            "U": "Percent"
          },
          {
            "N": "SystemMemUsage",
            "Sum": 10201.0,
            "U": "Megabytes"
          }
        ]
      },
      {
        "TS": 1602186483234,
        "NS": "GreengrassComponents",
        "M": [
          {
            "N": "NumberOfComponentsStopping",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsStarting",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsBroken",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsFinished",
            "Sum": 1.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsInstalled",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsRunning",
```

```
            "Sum": 7.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsNew",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsErrored",
            "Sum": 0.0,
            "U": "Count"
          },
          {
            "N": "NumberOfComponentsStateless",
            "Sum": 0.0,
            "U": "Count"
          }
        ]
      },
      {
        "TS": 1602186483234,
        "NS": "aws.greengrass.ClientDeviceAuth",
        "M": [
          {
            "N": "VerifyClientDeviceIdentity.Success",
            "Sum": 3.0,
            "U": "Count"
          },
          {
            "N": "VerifyClientDeviceIdentity.Failure",
            "Sum": 1.0,
            "U": "Count"
          },
          {
            "N": "AuthorizeClientDeviceActions.Success",
            "Sum": 20.0,
            "U": "Count"
          },
          {
            "N": "AuthorizeClientDeviceActions.Failure",
            "Sum": 5.0,
            "U": "Count"
          },
```

```
          {
            "N": "GetClientDeviceAuthToken.Success",
            "Sum": 5.0,
            "U": "Count"
          },
          {
            "N": "GetClientDeviceAuthToken.Failure",
            "Sum": 2.0,
            "U": "Count"
          },
          {
            "N": "SubscribeToCertificateUpdates.Success",
            "Sum": 10.0,
            "U": "Count"
          },
          {
            "N": "SubscribeToCertificateUpdates.Failure",
            "Sum": 1.0,
            "U": "Count"
          },
          {
            "N": "ServiceError",
            "Sum": 3.0,
            "U": "Count"
          }
        ]
      },
      {
        "TS": 1602186483234,
        "NS": "aws.greengrass.StreamManager",
        "M": [
          {
            "N": "BytesAppended",
            "Sum": 157745524.0,
            "U": "Bytes"
          },
          {
            "N": "BytesUploadedToIoTAnalytics",
            "Sum": 149012.0,
            "U": "Bytes"
          },
          {
            "N": "BytesUploadedToKinesis",
            "Sum": 12192.0,
```

```
            "U": "Bytes"
          },
          {
            "N": "BytesUploadedToIoTSiteWise",
            "Sum": 13321.0,
            "U": "Bytes"
          },
          {
            "N": "BytesUploadedToS3",
            "Sum": 12213.0,
            "U": "Bytes"
          }
        ]
      }
    ]
  }
}
```

L'ADParray contiene un elenco di punti dati aggregati con le seguenti proprietà:

`TS`

Il timestamp di quando i dati sono stati raccolti.

`NS`

Lo spazio dei nomi metrico.

`M`

L'elenco delle metriche. Una metrica contiene le seguenti proprietà:

`N`

Nome del parametro.

`Sum`

La somma dei valori della metrica in questo evento di telemetria.

`U`

L'unità del valore metrico.

Per ulteriori informazioni su ciascuna metrica, consulta. Metriche di telemetria

Prerequisiti per creare regole EventBridge

Prima di creare una EventBridge regola perAWS IoT Greengrass, è necessario effettuare le seguenti operazioni:

- Acquisisci familiarità con eventi, regole e obiettivi in. EventBridge
- Crea e configura gli obiettivi richiamati dalle tue regole. EventBridge Le regole possono richiamare molti tipi di destinazioni, ad esempio stream Amazon Kinesis, funzioniAWS Lambda, argomenti Amazon SNS e code Amazon SQS.

  La tua EventBridge regola e gli obiettivi associati devono trovarsi nello stesso luogo in Regione AWS cui hai creato le tue risorse Greengrass. Per ulteriori informazioni, consulta Service endpoints and quotas in. Riferimenti generali di AWS

Per ulteriori informazioni, consulta What is Amazon EventBridge? e Guida introduttiva ad Amazon EventBridge nella Amazon EventBridge User Guide.

Crea una regola di evento per ottenere dati di telemetria (console)

Utilizza i seguenti passaggi per AWS Management Console creare una EventBridge regola che riceva i dati di telemetria pubblicati dal dispositivo principale Greengrass. Ciò consente a server Web, indirizzi e-mail e altri sottoscrittori di argomenti di rispondere all'evento. Per ulteriori informazioni, consulta Creazione di una EventBridge regola che si attiva su un evento da una AWS risorsa nella Amazon EventBridge User Guide.

1. Apri la EventBridgeconsole Amazon e scegli Crea regola.
2. In Nome e descrizione, immettere un nome e una descrizione per la regola.
3. In Definisci modello, configurare il modello di regola.
   a. Scegli Event pattern (Modello di eventi).
   b. Scegliere Pre-defined pattern by service (Modello predefinito per servizio).
   c. Per Service provider (Provider di servizi), scegliere AWS.
   d. Per Nome servizio, scegliere Greengrass.
   e. Per Tipo di evento, seleziona Greengrass Telemetry Data.
4. In Seleziona bus eventi, mantenere le opzioni predefinite del bus eventi.
5. In Seleziona destinazioni, configura la tua destinazione. L'esempio seguente utilizza una coda Amazon SQS, ma puoi configurare altri tipi di destinazione.

a. Per Target, scegli la coda SQS.

b. Per Queue*, scegli la coda di destinazione.

6. In Tag - facoltativo, definire i tag per la regola o lasciare i campi vuoti.
7. Scegli Crea.

Crea una regola di evento per ottenere dati di telemetria (CLI)

Utilizza i seguenti passaggi per AWS CLI creare una EventBridge regola che riceva i dati di telemetria pubblicati dai dispositivi principali di Greengrass. Ciò consente a server Web, indirizzi e-mail e altri sottoscrittori di argomenti di rispondere all'evento.

1. Crea la regola.

   - Sostituisci *thing-name con il nome* dell'oggetto del dispositivo principale.

   Linux or Unix

   ```
   aws events put-rule \
     --name MyGreengrassTelemetryEventRule \
     --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail\": {\"ThingName
   \": [\"thing-name\"]}}"
   ```

   Windows Command Prompt (CMD)

   ```
   aws events put-rule ^
     --name MyGreengrassTelemetryEventRule ^
     --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail\": {\"ThingName
   \": [\"thing-name\"]}}"
   ```

   PowerShell

   ```
   aws events put-rule `
     --name MyGreengrassTelemetryEventRule `
     --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail\": {\"ThingName
   \": [\"thing-name\"]}}"
   ```

   Le proprietà omesse dal modello vengono ignorate.

2. Aggiungi l'argomento come destinazione della regola. L'esempio seguente utilizza Amazon SQS ma puoi configurare altri tipi di destinazione.

   - Sostituisci *queue-arn con* l'ARN della tua coda Amazon SQS.

   Linux or Unix

   ```
   aws events put-targets \
     --rule MyGreengrassTelemetryEventRule \
     --targets "Id"="1","Arn"="queue-arn"
   ```

   Windows Command Prompt (CMD)

   ```
   aws events put-targets ^
     --rule MyGreengrassTelemetryEventRule ^
     --targets "Id"="1","Arn"="queue-arn"
   ```

   PowerShell

   ```
   aws events put-targets `
     --rule MyGreengrassTelemetryEventRule `
     --targets "Id"="1","Arn"="queue-arn"
   ```

   > **Note**
   >
   > Per consentire EventBridge ad Amazon di richiamare la tua coda di destinazione, devi aggiungere una politica basata sulle risorse all'argomento. Per ulteriori informazioni, consulta le autorizzazioni di Amazon SQS nella Amazon EventBridge User Guide.

Per ulteriori informazioni, consulta la sezione Eventi e modelli di eventi EventBridge nella Amazon EventBridge User Guide.

Ricevi notifiche sullo stato di implementazione e integrità dei componenti

Le regole EventBridge degli eventi di Amazon ti forniscono notifiche sui cambiamenti di stato per le tue distribuzioni Greengrass ricevute dai tuoi dispositivi e per i componenti installati sul tuo dispositivo. EventBridge fornisce un flusso quasi in tempo reale di eventi di sistema che descrive i cambiamenti nelle risorse. AWS AWS IoT Greengrass invia questi eventi a EventBridge nel miglior modo possibile. Ciò significa che AWS IoT Greengrass tenta di inviare tutti gli eventi a EventBridge ma, in alcuni rari casi, un evento potrebbe non essere consegnato. Inoltre, è AWS IoT Greengrass possibile inviare più copie di un determinato evento, il che significa che gli ascoltatori dell'evento potrebbero non ricevere gli eventi nell'ordine in cui si sono verificati.

> **Note**
>
> Amazon EventBridge è un servizio di bus di eventi che puoi utilizzare per connettere le tue applicazioni con dati provenienti da una varietà di fonti, come i dispositivi principali Greengrass e le notifiche di distribuzione e componenti. Per ulteriori informazioni, consulta What is Amazon EventBridge? nella Amazon EventBridge User Guide.

Argomenti

Evento di modifica dello stato di implementazione

AWS IoT Greengrass emette un evento quando una distribuzione entra nei seguenti stati:`FAILED`, `SUCCEEDED` e`COMPLETED`. È possibile creare una EventBridge regola che venga eseguita per tutte le transizioni di stato o le transizioni verso gli stati specificati. Quando una distribuzione entra in uno stato che avvia una regola, EventBridge richiama le azioni target definite nella regola. In questo

modo è possibile inviare notifiche, acquisire informazioni sugli eventi, intraprendere azioni correttive o avviare altri eventi in risposta a una modifica dello stato. Ad esempio, è possibile creare regole per i seguenti casi d'uso:

- Avvia le operazioni successive all'implementazione, come il download delle risorse e la notifica al personale.
- Inviare notifiche in caso di distribuzione riuscita o non riuscita.
- Pubblicare parametri personalizzati sugli eventi di distribuzione.

L'evento per una modifica dello stato della distribuzione utilizza il formato seguente:

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass V2 Effective Deployment Status Change",
    "source":"aws.greengrass",
    "account":"123456789012",
    "region":"us-west-2",
    "time":"2018-03-22T00:38:11Z",
    "resources":["arn:aws:greengrass:us-
east-1:123456789012:coreDevices:MyGreengrassCore"],
    "detail":{
        "deploymentId": "4f38f1a7-3dd0-42a1-af48-EXAMPLE09681",
        "coreDeviceExecutionStatus": "FAILED|SUCCEEDED|COMPLETED",
        "statusDetails": {
          "errorStack": ["DEPLOYMENT_FAILURE", "ARTIFACT_DOWNLOAD_ERROR", "S3_ERROR",
 "S3_ACCESS_DENIED", "S3_HEAD_OBJECT_ACCESS_DENIED"],
          "errorTypes": ["DEPENDENCY_ERROR", "PERMISSION_ERROR"],
        },
        "reason": "S3_HEAD_OBJECT_ACCESS_DENIED: FAILED_NO_STATE_CHANGE: Failed to
 download artifact name: 's3://pentest27/nucleus/281/aws.greengrass.nucleus.zip' for
 component aws.greengrass.Nucleus-2.8.1, reason: S3 HeadObject returns 403 Access
 Denied. Ensure the IAM role associated with the core device has a policy granting
 s3:GetObject. null (Service: S3, Status Code: 403, Request ID: HR94ZNT2161DAR58,
 Extended Request ID: wTX4DDI+qigQt3uzwl9rlnQiYlBgwvPm/KJFWeFAn9t1mnGXTms/
luLCYANgq08RIH+x2H+hEKc=)"
     }
}
```

Puoi creare regole ed eventi che ti aggiorneranno sullo stato di una distribuzione. Un evento viene avviato quando una distribuzione viene completata come `FAILEDSUCCEEDED`, OR. `COMPLETED` Se

la distribuzione non è riuscita sul dispositivo principale, riceverai una risposta dettagliata che spiega perché la distribuzione non è riuscita. Per ulteriori informazioni sui codici di errore di distribuzione, consultaCodici di errore di distribuzione dettagliati.

Stati della distribuzione

- `FAILED`. La distribuzione non è riuscita.
- `SUCCEEDED`. La distribuzione destinata a un gruppo di oggetti è stata completata con successo.
- `COMPLETED`. L'implementazione mirata a un'operazione è stata completata con successo.

È possibile che gli eventi vengano duplicati o non funzionino. Per determinare l'ordine degli eventi, utilizza la proprietà `time`.

Per un elenco completo dei codici di errore in `errorStacks` and`errorTypes`, consulta Codici di errore di distribuzione dettagliati eCodici di stato dettagliati dei componenti.

Evento di modifica dello stato del componente

Per AWS IoT Greengrass le versioni 2.12.2 e precedenti, Greengrass emette un evento quando un componente entra nei seguenti stati: e. `ERRORED` `BROKEN` Per le versioni 2.12.3 e successive di Greengrass nucleus, Greengrass emette un evento quando un componente entra nei seguenti stati:,, e. `ERRORED` `BROKEN` `RUNNING` `FINISHED` Greengrass emetterà anche un evento al termine di un dispiegamento. È possibile creare una EventBridge regola che venga eseguita per tutte le transizioni di stato o le transizioni verso gli stati specificati. Quando un componente installato entra in uno stato che avvia una regola, EventBridge richiama le azioni di destinazione definite nella regola. In questo modo è possibile inviare notifiche, acquisire informazioni sugli eventi, intraprendere azioni correttive o avviare altri eventi in risposta a una modifica dello stato.

L'evento relativo alla modifica dello stato di un componente utilizza i seguenti formati:

Greengrass nucleus v2.12.2 and earlier

<title>Stato del componente: ERRORED o BROKEN</title>

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass V2 Installed Component Status Change",
    "source":"aws.greengrass",
```

```
    "account":"123456789012",
    "region":"us-west-2",
    "time":"2018-03-22T00:38:11Z",
    "resources":["arn:aws:greengrass:us-
east-1:123456789012:coreDevices:MyGreengrassCore"],
    "detail": {
       "components": [
          {
            "componentName": "MyComponent",
            "componentVersion": "1.0.0",
            "root": true,
            "lifecycleState": "ERRORED|BROKEN",
            "lifecycleStatusCodes": ["STARTUP_ERROR"],
            "lifecycleStateDetails": "An error occurred during startup. The startup
 script exited with code 1."
          }
        ]
    }
}
```

Greengrass nucleus v2.12.3 and later

<title>Stato del componente: ERRORED o BROKEN</title>

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass V2 Installed Component Status Change",
    "source":"aws.greengrass",
    "account":"123456789012",
    "region":"us-west-2",
    "time":"2018-03-22T00:38:11Z",
    "resources":["arn:aws:greengrass:us-
east-1:123456789012:coreDevices:MyGreengrassCore"],
    "detail": {
       "components": [
          {
            "componentName": "MyComponent",
            "componentVersion": "1.0.0",
            "root": true,
            "lifecycleState": "ERRORED|BROKEN",
            "lifecycleStatusCodes": ["STARTUP_ERROR"],
            "lifecycleStateDetails": "An error occurred during startup. The startup
 script exited with code 1."
```

```
            }
        ]
    }
}
```

<title>Stato del componente: RUNNING o FINISHED</title>

```
{
    "version":"0",
    "id":" cd4d811e-ab12-322b-8255-EXAMPLEb1bc8",
    "detail-type":"Greengrass V2 Installed Component Status Change",
    "source":"aws.greengrass",
    "account":"123456789012",
    "region":"us-west-2",
    "time":"2018-03-22T00:38:11Z",
    "resources":["arn:aws:greengrass:us-
east-1:123456789012:coreDevices:MyGreengrassCore"],
    "detail": {
       "components": [
          {
            "componentName": "MyComponent",
            "componentVersion": "1.0.0",
            "root": true,
            "lifecycleState": "RUNNING|FINISHED",
            "lifecycleStateDetails": null
          }
        ]
    }
}
```

Puoi creare regole ed eventi che ti aggiorneranno sullo stato di un componente installato. Un evento viene avviato quando un componente cambia stato sul dispositivo. Riceverai una risposta dettagliata che spiega perché un componente è danneggiato o danneggiato. Riceverai anche un codice di stato che indicherà il motivo dell'errore. Per ulteriori informazioni sui codici di stato dei componenti, vedereCodici di stato dettagliati dei componenti.

Prerequisiti per la creazione di regole EventBridge

Prima di creare una EventBridge regola per AWS IoT Greengrass, procedi come segue:

- Acquisisci familiarità con eventi, regole e obiettivi in. EventBridge

- Crea e configura gli obiettivi richiamati dalle tue regole. EventBridge Le regole possono richiamare molti tipi di target, tra cui:
  - Servizio di notifica semplice Amazon (Amazon Simple Notification Service (Amazon SNS))
  - AWS Lambda funzioni
  - Flusso di video Amazon Kinesis
  - Code di Amazon Simple Queue Service (Amazon SQS)

Per ulteriori informazioni, consulta What is Amazon EventBridge? e Guida introduttiva ad Amazon EventBridge nella Amazon EventBridge User Guide.

Configura le notifiche sullo stato del dispositivo (console)

Utilizza i seguenti passaggi per creare una EventBridge regola che pubblichi un argomento di Amazon SNS quando lo stato di distribuzione cambia per un gruppo. Ciò consente a server Web, indirizzi e-mail e altri sottoscrittori di argomenti di rispondere all'evento. Per ulteriori informazioni, consulta Creazione di una EventBridge regola che si attiva su un evento da una AWS risorsa nella Amazon EventBridge User Guide.

1. Apri la EventBridgeconsole Amazon.
2. Nel pannello di navigazione, scegli Regole.
3. Scegli Create rule (Crea regola).
4. Inserire un nome e una descrizione per la regola.

   Una regola non può avere lo stesso nome di un'altra regola nella stessa regione e sullo stesso router di eventi.
5. Per Select event bus (Seleziona bus di eventi), scegli il bus di eventi che desideri associare a questa regola. Se la regola deve cercare eventi corrispondenti provenienti dal tuo account, seleziona Bus di eventi predefiniti di AWS . Quando un AWS servizio del tuo account emette un evento, questo passa sempre al bus eventi predefinito del tuo account.
6. Per Rule type (Tipo di regola), scegli Rule with an event pattern (Regola con un modello di eventi).
7. Seleziona Successivo.
8. Per Event source (Origine eventi), seleziona AWS events (Eventi).
9. Per Event pattern, scegli AWS servizi.
10. Per l'AWS assistenza, scegli Greengrass.

11. Per il tipo di evento, scegli tra le seguenti opzioni:
    - Per gli eventi di implementazione, scegli Greengrass V2 Effective Deployment Status Change.
    - Per gli eventi relativi ai componenti, scegliete Greengrass V2 Installed Component Status Change.
12. Seleziona Successivo.
13. Per Target types (Tipi di destinazione), scegli AWS service (Servizio).
14. Per Seleziona una destinazione, configura la tua destinazione. Questo esempio utilizza un argomento Amazon SNS, ma puoi configurare altri tipi di destinazione per inviare notifiche.
    a. In Target (Destinazione), scegli SNS topic (Argomento SNS).
    b. Per Argomento, scegli l'argomento di destinazione.
    c. Seleziona Successivo.
15. Seleziona Successivo.
16. Rivedi i dettagli della regola e scegli Create rule (Crea regola).

Configurazione delle notifiche sullo stato del dispositivo (CLI)

Utilizza i seguenti passaggi per creare una EventBridge regola che pubblichi un argomento di Amazon SNS in caso di modifica dello stato di Greengrass. Ciò consente a server Web, indirizzi e-mail e altri sottoscrittori di argomenti di rispondere all'evento.

1. Crea la regola.
    - Per gli eventi di modifica dello stato di implementazione.

```
aws events put-rule \
  --name TestRule \
  --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail-type\":
 [\"Greengrass V2 Effective Deployment Status Change\"]}"
```

    - Per gli eventi di modifica dello stato dei componenti.

```
aws events put-rule \
  --name TestRule \
  --event-pattern "{\"source\": [\"aws.greengrass\"], \"detail-type\":
 [\"Greengrass V2 Installed Component Status Change\"]}"
```

Le proprietà omesse dal modello vengono ignorate.

2. Aggiungi l'argomento come destinazione della regola.

   - Sostituisci *topic-arn* con l'ARN del tuo argomento Amazon SNS.

```
aws events put-targets \
  --rule TestRule \
  --targets "Id"="1","Arn"="topic-arn"
```

> **Note**
>
> Per consentire EventBridge ad Amazon di indicare il tuo argomento di riferimento, devi aggiungere una politica basata sulle risorse all'argomento. Per ulteriori informazioni, consulta le autorizzazioni di Amazon SNS nella Amazon EventBridge User Guide.

Per ulteriori informazioni, consulta la sezione Eventi e modelli di eventi EventBridge nella Amazon EventBridge User Guide.

Configura le notifiche sullo stato del dispositivo (AWS CloudFormation)

Utilizza AWS CloudFormation i modelli per creare EventBridge regole che inviano notifiche sui cambiamenti di stato per le implementazioni del tuo gruppo Greengrass. Per ulteriori informazioni, consulta il riferimento ai tipi di EventBridge risorse Amazon nella Guida AWS CloudFormation per l'utente.

Consulta anche

- Verifica lo stato di distribuzione del dispositivo
- Che cos'è Amazon EventBridge? nella Amazon EventBridge User Guide

Controlla lo stato del dispositivo Greengrass core

I dispositivi core Greengrass segnalano lo stato dei loro componenti software a. AWS IoT GreengrassÈ possibile controllare il riepilogo dello stato di salute di ciascun dispositivo e controllare lo stato di ciascun componente su ciascun dispositivo.

I dispositivi principali hanno i seguenti stati di salute:

- `HEALTHY`— Il software AWS IoT Greengrass Core e tutti i componenti funzionano senza problemi sul dispositivo principale.
- `UNHEALTHY`— Il software AWS IoT Greengrass Core o un componente si trova in uno stato di errore sul dispositivo principale.

> **Note**
>
> AWS IoT Greengrass si affida ai singoli dispositivi per inviare aggiornamenti di stato a. Cloud AWS Se il software AWS IoT Greengrass Core non è in esecuzione sul dispositivo, o se il dispositivo non è connesso a Cloud AWS, lo stato riportato di quel dispositivo potrebbe non riflettere il suo stato attuale. Il timestamp dello stato indica quando lo stato del dispositivo è stato aggiornato l'ultima volta.
> I dispositivi principali inviano aggiornamenti di stato nei seguenti orari:
>
> - All'avvio del software AWS IoT Greengrass Core
> - Quando il dispositivo principale riceve una distribuzione dal Cloud AWS
> - Per Greengrass nucleus 2.12.2 e versioni precedenti, il dispositivo principale invia aggiornamenti di stato quando lo stato di qualsiasi componente sul dispositivo principale diventa o `ERRORED` `BROKEN`
> - Per Greengrass nucleus 2.12.3 e versioni successive, il dispositivo principale invia aggiornamenti di stato quando lo stato di un componente del dispositivo principale diventa,, o `ERRORED` `BROKEN` `RUNNING` `FINISHED`
> - A intervalli regolari configurabili, che per impostazione predefinita sono 24 ore
>
> Per AWS IoT Greengrass Core v2.7.0 e versioni successive, il dispositivo principale invia aggiornamenti di stato quando si verifica la distribuzione locale e la distribuzione nel cloud

Argomenti

Verifica lo stato di salute di un dispositivo principale

Puoi controllare lo stato dei singoli dispositivi principali.

Per verificare lo stato di un dispositivo principale (AWS CLI)

- Eseguite il comando seguente per recuperare lo stato di un dispositivo. Sostituisci *coreDeviceName* con il nome del dispositivo principale da interrogare.

  ```
  aws greengrassv2 get-core-device --core-device-thing-name coreDeviceName
  ```

  La risposta contiene informazioni sul dispositivo principale, incluso lo stato.

Verifica lo stato di un gruppo di dispositivi principale

È possibile controllare lo stato di un gruppo di dispositivi principali (un gruppo di oggetti).

Per controllare lo stato di un gruppo di dispositivi (AWS CLI)

- Esegui il comando seguente per recuperare lo stato di più dispositivi principali. Sostituire l'ARN nel comando con l'ARN del gruppo di cose da interrogare.

  ```
  aws greengrassv2 list-core-devices --thing-group-arn "arn:aws:iot:region:account-id:thinggroup/thingGroupName"
  ```

  La risposta contiene l'elenco dei dispositivi principali del gruppo di oggetti. Ogni voce dell'elenco contiene lo stato del dispositivo principale.

Controlla lo stato dei componenti del dispositivo principale

È possibile controllare lo stato, ad esempio lo stato del ciclo di vita, dei componenti software su un dispositivo principale. Per ulteriori informazioni sugli stati del ciclo di vita dei componenti, vedere. Sviluppa AWS IoT Greengrass componenti

Per controllare lo stato dei componenti su un dispositivo principale ()AWS CLI

- Eseguite il comando seguente per recuperare lo stato dei componenti su un dispositivo principale. Sostituisci *coreDeviceName*con il nome del dispositivo principale da interrogare.

  ```
  aws greengrassv2 list-installed-components --core-device-thing-name coreDeviceName
  ```

  La risposta contiene l'elenco dei componenti che funzionano sul dispositivo principale. Ogni voce dell'elenco contiene lo stato del ciclo di vita del componente, incluso lo stato attuale dei dati e l'ultima volta che il dispositivo principale Greengrass ha inviato un messaggio contenente un determinato componente al cloud. La risposta includerà anche la fonte di distribuzione più recente che ha portato il componente al dispositivo principale Greengrass.

  > **Note**
  >
  > Questo comando recupera un elenco impaginato dei componenti eseguiti da un dispositivo core Greengrass. Per impostazione predefinita, questo elenco non include i componenti che vengono distribuiti come dipendenze di altri componenti. È possibile includere dipendenze nella risposta impostando il parametro su. `topologyFilter` `ALL`

Esegui AWS Lambda funzioni

> **Note**
>
> AWS IoT Greengrassattualmente non supporta questa funzionalità sui dispositivi Windows core.

È possibile importare AWS Lambda funzioni come componenti eseguibili sui dispositivi AWS IoT Greengrass principali. È possibile eseguire questa operazione nei seguenti casi:

- Hai del codice applicativo nelle funzioni Lambda che desideri distribuire sui dispositivi principali.
- Hai applicazioni AWS IoT Greengrass V1 che desideri eseguire sui AWS IoT Greengrass V2 dispositivi principali. Per ulteriori informazioni, consulta Fase 2: Creare e distribuire componenti per migrare le applicazioni AWS IoT Greengrass V2AWS IoT Greengrass V1.

Le funzioni Lambda includono dipendenze dai seguenti componenti. Non è necessario definire questi componenti come dipendenze quando si importa la funzione. Quando si distribuisce il componente della funzione Lambda, la distribuzione include queste dipendenze dei componenti Lambda.

- Il componente di avvio Lambda (`aws.greengrass.LambdaLauncher`) gestisce i processi e la configurazione dell'ambiente.
- Il componente Lambda manager (`aws.greengrass.LambdaManager`) gestisce la comunicazione e la scalabilità tra processi.
- Il componente Lambda runtimes (`aws.greengrass.LambdaRuntimes`) fornisce artefatti per ogni runtime Lambda supportato.

Argomenti

Requisiti

I dispositivi principali e le funzioni Lambda devono soddisfare i seguenti requisiti per poter eseguire le funzioni sul software AWS IoT Greengrass Core:

- Il dispositivo principale deve soddisfare i requisiti per eseguire le funzioni Lambda. Se desideri che il dispositivo principale esegua funzioni Lambda containerizzate, il dispositivo deve soddisfare i requisiti per farlo. Per ulteriori informazioni, consulta Requisiti della funzione Lambda.
- È necessario installare i linguaggi di programmazione utilizzati dalla funzione Lambda sui dispositivi principali.

> **Tip**
>
> È possibile creare un componente che installa il linguaggio di programmazione e quindi specificare tale componente come dipendenza del componente della funzione Lambda. Greengrass supporta tutte le versioni supportate da Lambda dei runtime Python, Node.js e Java. Greengrass non applica alcuna restrizione aggiuntiva alle versioni di runtime Lambda obsolete. Puoi eseguire funzioni Lambda che utilizzano questi runtime obsoletiAWS IoT Greengrass, ma non puoi crearli in. AWS Lambda Per ulteriori informazioni sul AWS IoT Greengrass supporto per i runtime Lambda, consulta. Esegui AWS Lambda funzioni

Configura il ciclo di vita della funzione Lambda

Il ciclo di vita della funzione Greengrass Lambda determina quando una funzione viene avviata e come crea e utilizza i contenitori. Il ciclo di vita determina anche il modo in cui il software AWS IoT Greengrass Core conserva le variabili e la logica di preelaborazione esterne al gestore delle funzioni.

AWS IoT Greengrasssupporta cicli di vita on-demand (impostazione predefinita) e di lunga durata:

- Le funzioni su richiesta si avviano quando vengono richiamate e si interrompono quando non ci sono più attività da eseguire. Ogni chiamata della funzione crea un contenitore separato, chiamato anche sandbox, per elaborare le chiamate, a meno che un contenitore esistente non sia disponibile per il riutilizzo. Qualsiasi contenitore potrebbe elaborare i dati inviati alla funzione.

  È possibile eseguire più chiamate di una funzione su richiesta contemporaneamente.

Le variabili e la logica di preelaborazione definite al di fuori del gestore delle funzioni non vengono mantenute quando vengono creati nuovi contenitori.

- Le funzioni di lunga durata (o bloccate) iniziano all'avvio del software AWS IoT Greengrass Core e vengono eseguite in un singolo contenitore. Lo stesso contenitore elabora tutti i dati inviati alla funzione.

  Le chiamate multiple vengono messe in coda finché il software AWS IoT Greengrass Core non esegue le chiamate precedenti.

  Le variabili e la logica di preelaborazione definite all'esterno del gestore di funzioni vengono mantenute per ogni chiamata del gestore.

  Usa le funzioni Lambda di lunga durata quando devi iniziare a lavorare senza alcun input iniziale. Ad esempio, una funzione di lunga durata può caricare e avviare l'elaborazione di un modello di machine learning per essere pronta quando la funzione riceve i dati del dispositivo.

  > **Note**
  >
  > Le funzioni di lunga durata hanno dei timeout associati a ogni chiamata del relativo gestore. Se si desidera richiamare codice che viene eseguito all'infinito, è necessario avviarlo all'esterno del gestore. Assicuratevi che non vi sia alcun codice di blocco esterno al gestore che possa impedire l'inizializzazione della funzione.
  > Queste funzioni vengono eseguite a meno che il software AWS IoT Greengrass Core non si arresti, ad esempio durante una distribuzione o un riavvio. Queste funzioni non verranno eseguite se la funzione rileva un'eccezione non rilevata, supera i limiti di memoria o entra in uno stato di errore, ad esempio un timeout del gestore.

Per ulteriori informazioni sul riutilizzo dei container, consulta Understanding Container Reuse nel blog di Compute. AWS Lambda AWS

Configurare la containerizzazione delle funzioni Lambda

Per impostazione predefinita, le funzioni Lambda vengono eseguite all'interno di un AWS IoT Greengrass contenitore. I contenitori Greengrass garantiscono l'isolamento tra le funzioni e l'host. Questo isolamento aumenta la sicurezza sia per l'host che per le funzioni nel contenitore.

Ti consigliamo di eseguire le funzioni Lambda in un contenitore Greengrass, a meno che il tuo caso d'uso non richieda che vengano eseguite senza containerizzazione. Eseguendo le funzioni Lambda in un contenitore Greengrass, hai un maggiore controllo su come limitare l'accesso alle risorse.

È possibile eseguire una funzione Lambda senza containerizzazione nei seguenti casi:

- Vuoi eseguirlo AWS IoT Greengrass su un dispositivo che non supporta la modalità contenitore. Un esempio potrebbe essere se si desidera utilizzare una distribuzione Linux speciale o disporre di una versione precedente del kernel non aggiornata.
- Vuoi eseguire la tua funzione Lambda in un altro ambiente contenitore con il proprio OverlayFS, ma riscontri conflitti OverlayFS quando esegui in un contenitore Greengrass.
- È necessario accedere alle risorse locali con percorsi che non possono essere determinati al momento della distribuzione o i cui percorsi possono cambiare dopo la distribuzione. Un esempio di questa risorsa potrebbe essere un dispositivo collegabile.
- Hai un'applicazione precedente che è stata scritta come processo e riscontri problemi quando la esegui in un contenitore Greengrass.

Differenze nella containerizzazione

Containerizzazione	Note
Container Greengrass	• Tutte le AWS IoT Greengrass funzionalità sono disponibili quando si esegue una funzione Lambda in un contenitore Greengrass. • Le funzioni Lambda eseguite in un contenitore Greengrass non hanno accesso al codice distribuito di altre funzioni Lambda, anche se vengono eseguite con lo stesso gruppo di sistema. In altre parole, le funzioni Lambda vengono eseguite con un maggiore isolamento l'una dall'altra. • Poiché il software AWS IoT Greengrass Core esegue tutti i processi secondari nello stesso contenitore della funzione Lambda, i

Containerizzazione	Note
	processi secondari si interrompono quando si interrompe la funzione Lambda.

Nessun container	• Le seguenti funzionalità non sono disponibili per le funzioni Lambda non containerizzate:
	  • Limiti di memoria della funzione Lambda.
	  • Risorse volume e dispositivo locale. È necessario accedere a queste risorse utilizzando i relativi percorsi di file sul dispositivo principale anziché come risorse della funzione Lambda.
	• Se la funzione Lambda non containerizzata accede a una risorsa di machine learning, devi identificare il proprietario della risorsa e impostare le autorizzazioni di accesso sulla risorsa, non sulla funzione Lambda.
	• Le funzioni Lambda non containerizzate hanno accesso in sola lettura al codice distribuito di altre funzioni Lambda eseguite con lo stesso gruppo di sistema.

Se si modifica la containerizzazione di una funzione Lambda al momento della distribuzione, la funzione potrebbe non funzionare come previsto. Se la funzione Lambda utilizza risorse locali che non sono più disponibili con la nuova impostazione di containerizzazione, la distribuzione fallisce.

- Quando si modifica una funzione Lambda dall'esecuzione in un contenitore Greengrass all'esecuzione senza containerizzazione, i limiti di memoria della funzione vengono eliminati. È necessario accedere direttamente al file system anziché utilizzare le risorse locali collegate. È necessario rimuovere tutte le risorse collegate prima di distribuire la funzione Lambda.
- Quando si modifica una funzione Lambda dall'esecuzione senza containerizzazione all'esecuzione in un contenitore, la funzione Lambda perde l'accesso diretto al file system. È necessario definire un limite di memoria per ogni funzione o accettare il limite di memoria predefinito di 16 MB. Puoi configurare queste impostazioni per ogni funzione Lambda al momento della distribuzione.

Per modificare le impostazioni di containerizzazione per un componente della funzione Lambda, imposta il valore del parametro di `containerMode` configurazione su una delle seguenti opzioni quando distribuisci il componente.

- `NoContainer`— Il componente non viene eseguito in un ambiente di runtime isolato.
- `GreengrassContainer`— Il componente viene eseguito in un ambiente di runtime isolato all'interno del AWS IoT Greengrass contenitore.

Per ulteriori informazioni su come distribuire e configurare i componenti, vedere Implementazione AWS IoT Greengrass dei componenti sui dispositivi eAggiornamento delle configurazioni dei componenti.

Importazione di una funzione Lambda come componente (console)

Quando si utilizza la AWS IoT Greengrassconsole per creare un componente della funzione Lambda, si importa una AWS Lambda funzione esistente e quindi la si configura per creare un componente che funzioni sul dispositivo Greengrass.

Prima di iniziare, esamina i requisiti per eseguire le funzioni Lambda sui dispositivi Greengrass.

Attività

- Fase 1: Scegli una funzione Lambda da importare
- Fase 2: Configurazione dei parametri della funzione Lambda
- Fase 3: (Facoltativo) Specificare le piattaforme supportate per la funzione Lambda
- Fase 4: (Facoltativo) Specificare le dipendenze dei componenti per la funzione Lambda
- Fase 5: (Facoltativo) Eseguire la funzione Lambda in un contenitore
- Fase 6: Creare il componente della funzione Lambda

Fase 1: Scegli una funzione Lambda da importare

1. Nel menu di navigazione AWS IoT Greengrassdella console, scegli Componenti.
2. Nella pagina Componenti, scegli Crea componente.
3. Nella pagina Crea componente, in Informazioni sui componenti, scegli Importa funzione Lambda.
4. Nella funzione Lambda, cerca e scegli la funzione Lambda che desideri importare.

AWS IoT Greengrasscrea il componente con il nome della funzione Lambda.

5. Nella versione della funzione Lambda, scegli la versione da importare. Non puoi scegliere alias Lambda come. `$LATEST`

   AWS IoT Greengrasscrea il componente con la versione della funzione Lambda come versione semantica valida. Ad esempio, se la versione della funzione è `3`, la versione del componente diventa `3.0.0`.

Fase 2: Configurazione dei parametri della funzione Lambda

Nella pagina Crea componente, in Configurazione della funzione Lambda, configura i seguenti parametri da utilizzare per eseguire la funzione Lambda.

1. (Facoltativo) Aggiungi l'elenco delle fonti di eventi a cui la funzione Lambda si iscrive per i messaggi di lavoro. È possibile specificare le fonti di eventi per sottoscrivere questa funzione ai messaggi di pubblicazione/sottoscrizione locali e ai messaggi MQTT. AWS IoT Core La funzione Lambda viene chiamata quando riceve un messaggio da un'origine di eventi.

   > **Note**
   >
   > Per sottoscrivere questa funzione ai messaggi di altre funzioni o componenti Lambda, distribuisci il componente precedente del router di sottoscrizione quando distribuisci questo componente della funzione Lambda. Quando distribuisci il componente legacy del router di sottoscrizione, specifica gli abbonamenti utilizzati dalla funzione Lambda.

   In Origini di eventi, procedi come segue per aggiungere una fonte di eventi:

   a. Per ogni fonte di eventi che aggiungi, specifica le seguenti opzioni:

      - Argomento: l'argomento a cui sottoscrivere i messaggi.
      - Tipo: il tipo di origine dell'evento. Seleziona una delle opzioni seguenti:
        - Pubblicazione/sottoscrizione locale: sottoscrizione ai messaggi di pubblicazione/sottoscrizione locali.

Se si utilizza Greengrass nucleus v2.6.0 o versione successiva e Lambda manager v2.2.5 o versione successiva, è possibile utilizzare i caratteri jolly dell'argomento MQTT (and) nell'argomento quando si specifica questo tipo. **+ #**

- AWS IoT CoreAWS IoT CoreMQTT: sottoscrizione ai messaggi MQTT.

  È possibile utilizzare i caratteri jolly degli argomenti MQTT (+and#) nell'argomento quando si specifica questo tipo.

b. Per aggiungere un'altra fonte di eventi, scegliete Aggiungi origine evento e ripetete il passaggio precedente. Per rimuovere una fonte di evento, scegli Rimuovi accanto alla fonte dell'evento che desideri rimuovere.

2. Per Timeout (secondi), inserisci il periodo di tempo massimo, in secondi, che una funzione Lambda non bloccata può eseguire prima del timeout. Il valore predefinito è 3 secondi.
3. Per Bloccato, scegli se il componente della funzione Lambda è bloccato. L'impostazione predefinita è True.

   - Una funzione Lambda bloccata (o di lunga durata) si AWS IoT Greengrass avvia all'avvio e continua a funzionare nel proprio contenitore.
   - Una funzione Lambda non bloccata (o su richiesta) si avvia solo quando riceve un elemento di lavoro e esce dopo che è rimasto inattivo per un periodo di inattività massimo specificato. Se la funzione ha più elementi di lavoro, il software AWS IoT Greengrass Core crea più istanze della funzione.
4. (Facoltativo) In Parametri aggiuntivi, imposta i seguenti parametri della funzione Lambda.

   - Timeout di stato (secondi): l'intervallo in secondi in cui il componente della funzione Lambda invia aggiornamenti di stato al componente Lambda manager. Questo parametro si applica solo alle funzioni bloccate. Il valore predefinito è 60 secondi.
   - Dimensione massima della coda: la dimensione massima della coda dei messaggi per il componente della funzione Lambda. Il software AWS IoT Greengrass Core archivia i messaggi in una coda FIFO (first-in, first-out) fino a quando non può eseguire la funzione Lambda per consumare ogni messaggio. L'impostazione predefinita è 1.000 messaggi.
   - Numero massimo di istanze: il numero massimo di istanze che una funzione Lambda non bloccata può eseguire contemporaneamente. L'impostazione predefinita è 100 istanze.
   - Tempo di inattività massimo (secondi): il periodo di inattività massimo, in secondi, che una funzione Lambda non bloccata può rimanere inattiva prima che il software Core interrompa AWS IoT Greengrass il processo. Il valore predefinito è 60 secondi.

- Tipo di codifica: il tipo di payload supportato dalla funzione Lambda. Seleziona una delle opzioni seguenti:
  - JSON
  - Binary

  Il valore predefinito è JSON.

5. (Facoltativo) Specificate l'elenco degli argomenti della riga di comando da passare alla funzione Lambda durante l'esecuzione.

   a. In Parametri aggiuntivi, Elabora argomenti, scegli Aggiungi argomento.

   b. Per ogni argomento che aggiungi, inserisci l'argomento che desideri passare alla funzione.

   c. Per rimuovere un argomento, scegliete Rimuovi accanto all'argomento che desiderate rimuovere.

6. (Facoltativo) Specificate le variabili di ambiente disponibili per la funzione Lambda quando viene eseguita. Le variabili di ambiente consentono di memorizzare e aggiornare le impostazioni di configurazione senza la necessità di modificare il codice della funzione.

   a. In Parametri aggiuntivi, Variabili di ambiente, scegli Aggiungi variabile di ambiente.

   b. Per ogni variabile di ambiente che aggiungi, specifica le seguenti opzioni:

      - Chiave: il nome della variabile.
      - Valore: il valore predefinito per questa variabile.

   c. Per rimuovere una variabile di ambiente, scegli Rimuovi accanto alla variabile di ambiente che desideri rimuovere.

Fase 3: (Facoltativo) Specificare le piattaforme supportate per la funzione Lambda

Tutti i dispositivi principali hanno attributi per il sistema operativo e l'architettura. Quando si distribuisce il componente della funzione Lambda, AWS IoT Greengrass il software Core confronta i valori della piattaforma specificati con gli attributi della piattaforma sul dispositivo principale per determinare se la funzione Lambda è supportata su quel dispositivo.

> **Note**
>
> È inoltre possibile specificare attributi di piattaforma personalizzati quando si distribuisce il componente Greengrass nucleus su un dispositivo principale. Per ulteriori informazioni, consulta il parametro platform overrides del componente Greengrass nucleus.

In Configurazione della funzione Lambda, Parametri aggiuntivi, Piattaforme, procedi come segue per specificare le piattaforme supportate da questa funzione Lambda.

1. Per ogni piattaforma, specifica le seguenti opzioni:
   - Sistema operativo: il nome del sistema operativo per la piattaforma. Attualmente, l'unico valore supportato è `linux`.
   - Architettura: l'architettura del processore per la piattaforma. I valori supportati sono:
     - `amd64`
     - `arm`
     - `aarch64`
     - `x86`
2. Per aggiungere un'altra piattaforma, scegli Aggiungi piattaforma e ripeti il passaggio precedente. Per rimuovere una piattaforma supportata, scegli Rimuovi accanto alla piattaforma che desideri rimuovere.

Fase 4: (Facoltativo) Specificare le dipendenze dei componenti per la funzione Lambda

Le dipendenze tra i componenti identificano i componenti aggiuntivi AWS forniti o i componenti personalizzati utilizzati dalla funzione. Quando si distribuisce il componente della funzione Lambda, la distribuzione include queste dipendenze per l'esecuzione della funzione.

> **Important**
>
> Per importare una funzione Lambda creata per l'esecuzione su AWS IoT Greengrass V1, è necessario definire le dipendenze dei singoli componenti per le funzionalità utilizzate dalla funzione, come segreti, ombre locali e gestore di flussi. Definisci questi componenti

come dipendenze rigide in modo che il componente della funzione Lambda si riavvii se la dipendenza cambia stato. Per ulteriori informazioni, consulta Importa funzioni Lambda V1.

In Configurazione della funzione Lambda, Parametri aggiuntivi, Dipendenze dei componenti, completa i seguenti passaggi per specificare le dipendenze dei componenti per la tua funzione Lambda.

1. Scegli Aggiungi dipendenza.
2. Per ogni dipendenza dal componente che aggiungi, specifica le seguenti opzioni:
   - Nome del componente: il nome del componente. Ad esempio, immettete **`aws.greengrass.StreamManager`** per includere il componente stream manager.
   - Requisito di versione: il vincolo di versione semantica in stile npm che identifica le versioni compatibili di questa dipendenza dal componente. È possibile specificare una singola versione o un intervallo di versioni. Ad esempio, immettere **`^1.0.0`** per specificare che questa funzione Lambda dipende da qualsiasi versione della prima versione principale del componente stream manager. Per ulteriori informazioni sui vincoli di versione semantica, consulta il calcolatore semver npm.
   - Tipo: il tipo di dipendenza. Seleziona una delle opzioni seguenti:
     - Difficile: il componente della funzione Lambda si riavvia se la dipendenza cambia stato. Questa è la selezione predefinita.
     - Soft: il componente della funzione Lambda non si riavvia se la dipendenza cambia stato.
3. Per rimuovere una dipendenza da un componente, scegli Rimuovi accanto alla dipendenza del componente

Fase 5: (Facoltativo) Eseguire la funzione Lambda in un contenitore

Per impostazione predefinita, le funzioni Lambda vengono eseguite in un ambiente di runtime isolato all'interno del software AWS IoT Greengrass Core. Puoi anche scegliere di eseguire la funzione Lambda come processo senza alcun isolamento (ovvero in modalità No container).

In Configurazione del processo Linux, per la modalità Isolamento, scegli tra le seguenti opzioni per selezionare la containerizzazione per la tua funzione Lambda:

- Contenitore Greengrass: la funzione Lambda viene eseguita in un contenitore. Questa è la selezione predefinita.

- Nessun contenitore: la funzione Lambda viene eseguita come processo senza alcun isolamento.

Se esegui la funzione Lambda in un contenitore, completa i seguenti passaggi per configurare la configurazione del processo per la funzione Lambda.

1. Configura la quantità di memoria e le risorse di sistema, come volumi e dispositivi, da rendere disponibili al contenitore.

   In Parametri del contenitore, procedi come segue.

   a. In Dimensione della memoria, inserisci la dimensione della memoria che desideri allocare al contenitore. È possibile specificare la dimensione della memoria in MB o kB.

   b. Per la cartella sys di sola lettura, scegliete se il contenitore può leggere o meno le informazioni dalla cartella del dispositivo. `/sys` L'impostazione predefinita è False.

2. (Facoltativo) Configura i volumi locali a cui può accedere la funzione Lambda containerizzata. Quando definisci un volume, il software AWS IoT Greengrass Core monta i file di origine sulla destinazione all'interno del container.

   a. In Volumi, scegli Aggiungi volume.

   b. Per ogni volume che aggiungi, specifica le seguenti opzioni:

      - Volume fisico: il percorso della cartella di origine sul dispositivo principale.
      - Volume logico: il percorso della cartella di destinazione nel contenitore.
      - Autorizzazione: (Facoltativo) L'autorizzazione ad accedere alla cartella di origine dal contenitore. Seleziona una delle opzioni seguenti:
        - Sola lettura: la funzione Lambda ha accesso in sola lettura alla cartella di origine. Questa è la selezione predefinita.
        - Lettura/scrittura: la funzione Lambda ha accesso in lettura/scrittura alla cartella di origine.
      - Aggiungi proprietario del gruppo - (Facoltativo) Se aggiungere o meno il gruppo di sistema che esegue il componente della funzione Lambda come proprietario della cartella di origine. L'impostazione predefinita è False.

   c. Per rimuovere un volume, scegli Rimuovi accanto al volume che desideri rimuovere.

3. (Facoltativo) Configura i dispositivi del sistema locale a cui può accedere la funzione Lambda containerizzata.

a. In Dispositivi, scegli Aggiungi dispositivo.

b. Per ogni dispositivo che aggiungi, specifica le seguenti opzioni:

- Percorso di montaggio: il percorso verso il dispositivo di sistema sul dispositivo principale.
- Autorizzazione: (Facoltativo) L'autorizzazione ad accedere al dispositivo di sistema dal contenitore. Seleziona una delle opzioni seguenti:
  - Sola lettura: la funzione Lambda ha accesso in sola lettura al dispositivo di sistema. Questa è la selezione predefinita.
  - Lettura/scrittura: la funzione Lambda ha accesso in lettura/scrittura alla cartella di origine.
- Aggiungi proprietario del gruppo - (Facoltativo) Se aggiungere o meno il gruppo di sistema che esegue il componente della funzione Lambda come proprietario del dispositivo di sistema. L'impostazione predefinita è False.

Fase 6: Creare il componente della funzione Lambda

Dopo aver configurato le impostazioni per il componente della funzione Lambda, scegli Crea per completare la creazione del nuovo componente.

Per eseguire la funzione Lambda sul dispositivo principale, è quindi possibile distribuire il nuovo componente sui dispositivi principali. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.

Importa una funzione Lambda come componente () AWS CLI

Usa l'CreateComponentVersionoperazione per creare componenti dalle funzioni Lambda. Quando chiamate questa operazione, specificate di `lambdaFunction` importare una funzione Lambda.

Attività

- Fase 1: Definire la configurazione della funzione Lambda
- Fase 2: Creare il componente della funzione Lambda

Fase 1: Definire la configurazione della funzione Lambda

1. Create un file chiamato`lambda-function-component.json`, quindi copiate il seguente oggetto JSON nel file. Sostituisci `lambdaArn` con l'ARN della funzione Lambda da importare.

   ```
   {
     "lambdaFunction": {
       "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1"
     }
   }
   ```

   > **Important**
   >
   > È necessario specificare un ARN che includa la versione della funzione da importare. Non è possibile utilizzare alias di versione come `$LATEST`.

2. (Facoltativo) Specificate il nome (`componentName`) del componente. Se ometti questo parametro, AWS IoT Greengrass crea il componente con il nome della funzione Lambda.

   ```
   {
     "lambdaFunction": {
       "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
       "componentName": "com.example.HelloWorldLambda"
     }
   }
   ```

3. (Facoltativo) Specificate la versione (`componentVersion`) per il componente. Se ometti questo parametro, AWS IoT Greengrass crea il componente con la versione della funzione Lambda come versione semantica valida. Ad esempio, se la versione della funzione è `3`, la versione del componente diventa `3.0.0`.

   > **Note**
   >
   > Ogni versione del componente che carichi deve essere unica. Assicurati di caricare la versione corretta del componente, perché non puoi modificarla dopo averla caricata. AWS IoT Greengrassutilizza versioni semantiche per i componenti. Le versioni semantiche seguono una delle principali. minore. sistema di numerazione delle patch. Ad

esempio, la versione `1.0.0` rappresenta la prima release principale di un componente. Per ulteriori informazioni, consultate la specifica della versione semantica.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0"
  }
}
```

4. (Facoltativo) Specificate le piattaforme supportate da questa funzione Lambda. Ogni piattaforma contiene una mappa di attributi che identificano una piattaforma. Tutti i dispositivi principali hanno attributi per il sistema operativo (`os`) e l'architettura (`architecture`). Il software AWS IoT Greengrass Core può aggiungere altri attributi della piattaforma. È inoltre possibile specificare attributi di piattaforma personalizzati quando si distribuisce il componente Greengrass nucleus su un dispositivo principale. Esegui questa operazione:

   a. Aggiungi un elenco di piattaforme (`componentPlatforms`) alla funzione Lambda in. `lambda-function-component.json`

   ```
   {
     "lambdaFunction": {
       "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
       "componentName": "com.example.HelloWorldLambda",
       "componentVersion": "1.0.0",
       "componentPlatforms": [

       ]
     }
   }
   ```

   b. Aggiungi ogni piattaforma supportata all'elenco. Ogni piattaforma dispone di un'interfaccia intuitiva `name` per identificarla e di una mappa di attributi. L'esempio seguente specifica che questa funzione supporta i dispositivi x86 che eseguono Linux.

   ```
   {
     "name": "Linux x86",
     "attributes": {
   ```

```
    "os": "linux",
    "architecture": "x86"
  }
}
```

È `lambda-function-component.json` possibile che contenga un documento simile all'esempio seguente.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ]
  }
}
```

5. (Facoltativo) Specificate le dipendenze dei componenti per la funzione Lambda. Quando si distribuisce il componente della funzione Lambda, la distribuzione include queste dipendenze per l'esecuzione della funzione.

> **Important**
>
> Per importare una funzione Lambda creata per l'esecuzione su AWS IoT Greengrass V1, è necessario definire le dipendenze dei singoli componenti per le funzionalità utilizzate dalla funzione, come segreti, ombre locali e gestore di flussi. Definisci questi componenti come dipendenze rigide in modo che il componente della funzione Lambda si riavvii se la dipendenza cambia stato. Per ulteriori informazioni, consulta Importa funzioni Lambda V1.

Esegui questa operazione:

a. Aggiungi una mappa delle dipendenze dei componenti (`componentDependencies`) alla funzione Lambda in. `lambda-function-component.json`

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {

    }
  }
}
```

b. Aggiungi la dipendenza di ogni componente alla mappa. Specificate il nome del componente come chiave e specificate un oggetto con i seguenti parametri:

- `versionRequirement`— Il vincolo di versione semantica in stile npm che identifica le versioni compatibili della dipendenza del componente. È possibile specificare una singola versione o un intervallo di versioni. Per ulteriori informazioni sui vincoli di versione semantica, consulta il calcolatore semver npm.
- `dependencyType`— (Facoltativo) Il tipo di dipendenza. Scegli tra le seguenti opzioni:
  - `SOFT`— Il componente della funzione Lambda non si riavvia se la dipendenza cambia stato.
  - `HARD`— Il componente della funzione Lambda si riavvia se la dipendenza cambia stato.

  Il valore predefinito è `HARD`.

L'esempio seguente specifica che questa funzione Lambda dipende da qualsiasi versione della prima versione principale del componente stream manager. Il componente della funzione Lambda si riavvia quando lo stream manager si riavvia o si aggiorna.

```
{
  "aws.greengrass.StreamManager": {
    "versionRequirement": "^1.0.0",
    "dependencyType": "HARD"
  }
}
```

È `lambda-function-component.json` possibile che contenga un documento simile all'esempio seguente.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    }
  }
}
```

6. (Facoltativo) Configura i parametri della funzione Lambda da utilizzare per eseguire la funzione. È possibile configurare opzioni quali variabili di ambiente, origini degli eventi dei messaggi, timeout e impostazioni del contenitore. Esegui questa operazione:

a. Aggiungi l'oggetto dei parametri Lambda (`componentLambdaParameters`) alla funzione Lambda in. `lambda-function-component.json`

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {

    }
  }
}
```

b. (Facoltativo) Specificate le fonti di eventi a cui la funzione Lambda si iscrive per i messaggi di lavoro. È possibile specificare le fonti di eventi per sottoscrivere questa funzione ai messaggi di pubblicazione/sottoscrizione locali e ai messaggi MQTT. AWS IoT Core La funzione Lambda viene chiamata quando riceve un messaggio da un'origine di eventi.

> **Note**
>
> Per sottoscrivere questa funzione ai messaggi di altre funzioni o componenti Lambda, distribuisci il componente precedente del router di sottoscrizione quando distribuisci questo componente della funzione Lambda. Quando distribuisci il

componente legacy del router di sottoscrizione, specifica gli abbonamenti utilizzati dalla funzione Lambda.

Esegui questa operazione:

i. Aggiungi l'elenco delle fonti di eventi (`eventSources`) ai parametri della funzione Lambda.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [

      ]
    }
  }
}
```

ii. Aggiungi ogni fonte di eventi all'elenco. Ogni fonte di eventi ha i seguenti parametri:

- `topic`— L'argomento a cui sottoscrivere i messaggi.
- `type`— Il tipo di origine dell'evento. Seleziona una delle opzioni seguenti:

- `PUB_SUB`: iscriviti ai messaggi di pubblicazione/sottoscrizione locali.

  Se si utilizza Greengrass nucleus v2.6.0 o versione successiva e Lambda manager v2.2.5 o versione successiva, è possibile utilizzare i caratteri jolly + (and) dell'argomento MQTT quando si specifica questo tipo. `# topic`

- `IOT_CORE`: sottoscrivi i messaggi MQTT di AWS IoT Core.

  È possibile utilizzare i caratteri jolly (e) dell'argomento MQTT quando si specifica questo tipo. `+ # topic`

L'esempio seguente sottoscrive AWS IoT Core MQTT su argomenti che corrispondono al filtro degli argomenti. `hello/world/+`

```
{
  "topic": "hello/world/+",
  "type": "IOT_CORE"
}
```

Il tuo `lambda-function-component.json` potrebbe essere simile all'esempio seguente.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
```

```
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ]
    }
  }
}
```

c. (Facoltativo) Specificate uno dei seguenti parametri nell'oggetto parametri della funzione Lambda:

- `environmentVariables`— La mappa delle variabili di ambiente disponibili per la funzione Lambda durante l'esecuzione.
- `execArgs`— L'elenco di argomenti da passare alla funzione Lambda durante l'esecuzione.
- `inputPayloadEncodingType`— Il tipo di payload supportato dalla funzione Lambda. Seleziona una delle opzioni seguenti:
  - `json`
  - `binary`

  Impostazione predefinita: `json`
- `pinned`— Se la funzione Lambda è bloccata o meno. Il valore predefinito è `true`.
  - Una funzione Lambda bloccata (o di lunga durata) si AWS IoT Greengrass avvia all'avvio e continua a funzionare nel proprio contenitore.
  - Una funzione Lambda non bloccata (o su richiesta) si avvia solo quando riceve un elemento di lavoro e esce dopo che è rimasto inattivo per un periodo di inattività massimo specificato. Se la funzione ha più elementi di lavoro, il software AWS IoT Greengrass Core crea più istanze della funzione.

  `maxIdleTimeInSeconds`Da utilizzare per impostare il tempo di inattività massimo per la funzione.
- `timeoutInSeconds`— Il periodo di tempo massimo, in secondi, che la funzione Lambda può eseguire prima del timeout. Il valore predefinito è 3 secondi.

- `statusTimeoutInSeconds`— L'intervallo in secondi con cui il componente della funzione Lambda invia gli aggiornamenti di stato al componente Lambda manager. Questo parametro si applica solo alle funzioni bloccate. Il valore predefinito è 60 secondi.
- `maxIdleTimeInSeconds`— Il tempo massimo, in secondi, in cui una funzione Lambda non bloccata può rimanere inattiva prima che il software AWS IoT Greengrass Core interrompa il processo. Il valore predefinito è 60 secondi.
- `maxInstancesCount`— Il numero massimo di istanze che una funzione Lambda non bloccata può eseguire contemporaneamente. L'impostazione predefinita è 100 istanze.
- `maxQueueSize`— La dimensione massima della coda di messaggi per il componente della funzione Lambda. Il software AWS IoT Greengrass Core archivia i messaggi in una coda FIFO (first-in-first-out) fino a quando non può eseguire la funzione Lambda per consumare ogni messaggio. L'impostazione predefinita è 1.000 messaggi.

È `lambda-function-component.json` possibile che contenga un documento simile al seguente esempio.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
```

```
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500
    }
  }
}
```

d. (Facoltativo) Configura le impostazioni del contenitore per la funzione Lambda. Per impostazione predefinita, le funzioni Lambda vengono eseguite in un ambiente di runtime isolato all'interno del software AWS IoT Greengrass Core. Puoi anche scegliere di eseguire la funzione Lambda come processo senza alcun isolamento. Se si esegue la funzione Lambda in un contenitore, si configura la dimensione della memoria del contenitore e le risorse di sistema disponibili per la funzione Lambda. Esegui questa operazione:

   i. Aggiungi l'oggetto dei parametri del processo Linux (`linuxProcessParams`) all'oggetto parametri Lambda in. `lambda-function-component.json`

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
```

```
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {

      }
    }
  }
}
```

ii. (Facoltativo) Specificate se la funzione Lambda viene eseguita o meno in un contenitore. Aggiungete il `isolationMode` parametro all'oggetto dei parametri di processo e scegliete una delle seguenti opzioni:

- `GreengrassContainer`— La funzione Lambda viene eseguita in un contenitore.
- `NoContainer`— La funzione Lambda viene eseguita come processo senza alcun isolamento.

Il valore predefinito è `GreengrassContainer`.

iii. (Facoltativo) Se si esegue la funzione Lambda in un contenitore, è possibile configurare la quantità di memoria e le risorse di sistema, come volumi e dispositivi, da rendere disponibili al contenitore. Esegui questa operazione:

A. Aggiungete l'oggetto dei parametri del contenitore (`containerParams`) all'oggetto dei parametri del processo Linux in`lambda-function-component.json`.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
```

```
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {
        "containerParams": {

        }
      }
    }
  }
}
```

B. (Facoltativo) Aggiungete il `memorySizeInKB` parametro per specificare la dimensione della memoria del contenitore. L'impostazione predefinita è 16.384 KB (16 MB).

C. (Facoltativo) Aggiungi il `mountROSysfs` parametro per specificare se il contenitore può leggere o meno le informazioni dalla cartella del `/sys` dispositivo. Il valore predefinito è `false`.

D. (Facoltativo) Configura i volumi locali a cui può accedere la funzione Lambda containerizzata. Quando definisci un volume, il software AWS IoT Greengrass Core monta i file di origine sulla destinazione all'interno del container. Esegui questa operazione:

   I. Aggiungi l'elenco dei volumi (`volumes`) ai parametri del contenitore.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
```

```
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {
        "containerParams": {
          "memorySizeInKB": 32768,
          "mountROSysfs": true,
          "volumes": [

          ]
        }
      }
    }
  }
}
```

II. Aggiungi ogni volume all'elenco. Ogni volume presenta i seguenti parametri:

- `sourcePath`— Il percorso della cartella di origine sul dispositivo principale.
- `destinationPath`— Il percorso della cartella di destinazione nel contenitore.
- `permission`— (Facoltativo) L'autorizzazione ad accedere alla cartella di origine dal contenitore. Seleziona una delle opzioni seguenti:
  - `ro`— La funzione Lambda ha accesso in sola lettura alla cartella di origine.
  - `rw`— La funzione Lambda ha accesso in lettura/scrittura alla cartella di origine.

  Il valore predefinito è `ro`.
- `addGroupOwner`— (Facoltativo) Se aggiungere o meno il gruppo di sistema che esegue il componente della funzione Lambda come proprietario della cartella di origine. Il valore predefinito è `false`.

È `lambda-function-component.json` possibile che contenga un documento simile al seguente esempio.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
```

```
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {
        "containerParams": {
          "memorySizeInKB": 32768,
          "mountROSysfs": true,
          "volumes": [
            {
              "sourcePath": "/var/data/src",
              "destinationPath": "/var/data/dest",
              "permission": "rw",
              "addGroupOwner": true
            }
          ]
        }
      }
    }
  }
}
```

E. (Facoltativo) Configura i dispositivi del sistema locale a cui può accedere la funzione Lambda containerizzata. Esegui questa operazione:

I. Aggiungi l'elenco dei dispositivi di sistema (`devices`) ai parametri del contenitore.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
```

```
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {
        "containerParams": {
          "memorySizeInKB": 32768,
          "mountROSysfs": true,
          "volumes": [
            {
              "sourcePath": "/var/data/src",
              "destinationPath": "/var/data/dest",
              "permission": "rw",
              "addGroupOwner": true
            }
          ],
          "devices": [

          ]
        }
      }
    }
  }
}
```

II. Aggiungi ogni dispositivo di sistema all'elenco. Ogni dispositivo di sistema presenta i seguenti parametri:

- `path`— Il percorso verso il dispositivo di sistema sul dispositivo principale.
- `permission`— (Facoltativo) L'autorizzazione ad accedere al dispositivo di sistema dal contenitore. Seleziona una delle opzioni seguenti:
  - `ro`— La funzione Lambda ha accesso in sola lettura al dispositivo di sistema.
  - `rw`— La funzione Lambda ha accesso in lettura/scrittura al dispositivo di sistema.

  Il valore predefinito è `ro`.
- `addGroupOwner`— (Facoltativo) Se aggiungere o meno il gruppo di sistema che esegue il componente della funzione Lambda come proprietario del dispositivo di sistema. Il valore predefinito è `false`.

Il tuo `lambda-function-component.json` potrebbe contenere un documento simile al seguente esempio.

```
{
  "lambdaFunction": {
    "lambdaArn": "arn:aws:lambda:region:account-
id:function:HelloWorld:1",
    "componentName": "com.example.HelloWorldLambda",
    "componentVersion": "1.0.0",
    "componentPlatforms": [
      {
        "name": "Linux x86",
        "attributes": {
          "os": "linux",
          "architecture": "x86"
        }
      }
    ],
    "componentDependencies": {
      "aws.greengrass.StreamManager": {
        "versionRequirement": "^1.0.0",
        "dependencyType": "HARD"
      }
    },
    "componentLambdaParameters": {
      "eventSources": [
        {
          "topic": "hello/world/+",
          "type": "IOT_CORE"
        }
      ],
      "environmentVariables": {
        "LIMIT": "300"
      },
      "execArgs": [
        "-d"
      ],
      "inputPayloadEncodingType": "json",
      "pinned": true,
      "timeoutInSeconds": 120,
      "statusTimeoutInSeconds": 30,
      "maxIdleTimeInSeconds": 30,
```

```
      "maxInstancesCount": 50,
      "maxQueueSize": 500,
      "linuxProcessParams": {
        "containerParams": {
          "memorySizeInKB": 32768,
          "mountROSysfs": true,
          "volumes": [
            {
              "sourcePath": "/var/data/src",
              "destinationPath": "/var/data/dest",
              "permission": "rw",
              "addGroupOwner": true
            }
          ],
          "devices": [
            {
              "path": "/dev/sda3",
              "permission": "rw",
              "addGroupOwner": true
            }
          ]
        }
      }
    }
  }
}
```

7. (Facoltativo) Aggiungi tag (`tags`) per il componente. Per ulteriori informazioni, consulta Tagging delle risorse AWS IoT Greengrass Version 2..

Fase 2: Creare il componente della funzione Lambda

1. Esegui il comando seguente per creare il componente della funzione Lambda da. `lambda-function-component.json`

   ```
   aws greengrassv2 create-component-version --cli-input-json file://lambda-function-component.json
   ```

   La risposta è simile all'esempio seguente se la richiesta ha esito positivo.

   ```
   {
   ```

```
  "arn":
 "arn:aws:greengrass:region:123456789012:components:com.example.HelloWorldLambda:versions:1
  "componentName": "com.example.HelloWorldLambda",
  "componentVersion": "1.0.0",
  "creationTimestamp": "Mon Dec 15 20:56:34 UTC 2020",
  "status": {
    "componentState": "REQUESTED",
    "message": "NONE",
    "errors": {}
  }
}
```

Copia il file `arn` dall'output per verificare lo stato del componente nel passaggio successivo.

2. Quando si crea un componente, il suo stato è`REQUESTED`. Quindi, AWS IoT Greengrass verifica che il componente sia implementabile. È possibile eseguire il comando seguente per interrogare lo stato del componente e verificare che il componente sia distribuibile. Sostituisci `arn` con l'ARN del passaggio precedente.

   ```
   aws greengrassv2 describe-component \
     --arn "arn:aws:greengrass:region:account-
   id:components:com.example.HelloWorldLambda:versions:1.0.0"
   ```

   Se il componente viene convalidato, la risposta indica che lo stato del componente è. `DEPLOYABLE`

   ```
   {
     "arn": "arn:aws:greengrass:region:account-
   id:components:com.example.HelloWorldLambda:versions:1.0.0",
     "componentName": "com.example.HelloWorldLambda",
     "componentVersion": "1.0.0",
     "creationTimestamp": "2020-12-15T20:56:34.376000-08:00",
     "publisher": "AWS Lambda",
     "status": {
       "componentState": "DEPLOYABLE",
       "message": "NONE",
       "errors": {}
     },
     "platforms": [
       {
         "name": "Linux x86",
         "attributes": {
   ```

```
        "architecture": "x86",
        "os": "linux"
      }
    }
  ]
}
```

Dopo aver installato il componenteDEPLOYABLE, puoi distribuire la funzione Lambda sui tuoi dispositivi principali. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.

Usa il SDK per dispositivi AWS IoT per comunicare con il nucleo Greengrass, altri componenti e AWS IoT Core

I componenti in esecuzione sul dispositivo principale possono utilizzare la libreria AWS IoT Greengrass Core interprocess communication (IPC) SDK per dispositivi AWS IoT per comunicare con il AWS IoT Greengrass nucleo e altri componenti Greengrass. Per sviluppare ed eseguire componenti personalizzati che utilizzano IPC, è necessario utilizzare il servizio IPC SDK per dispositivi AWS IoT di AWS IoT Greengrass base ed eseguire operazioni IPC.

L'interfaccia IPC supporta due tipi di operazioni:

- Richiesta/risposta

  I componenti inviano una richiesta al servizio IPC e ricevono una risposta che contiene il risultato della richiesta.

- Subscription

  I componenti inviano una richiesta di sottoscrizione al servizio IPC e prevedono un flusso di messaggi relativi agli eventi in risposta. I componenti forniscono un gestore di sottoscrizioni che gestisce i messaggi di eventi, gli errori e la chiusura dei flussi. SDK per dispositivi AWS IoTInclude un'interfaccia di gestione con la risposta e i tipi di eventi corretti per ogni operazione IPC. Per ulteriori informazioni, consulta Iscriviti ai flussi di eventi IPC.

Argomenti

- Interazione con la configurazione dei componenti
- Recupera valori segreti
- Interagisci con le ombre locali
- Gestisci le implementazioni e i componenti locali
- Autentica e autorizza i dispositivi client

Versioni client IPC

Nelle versioni successive degli SDK Java e Python, AWS IoT Greengrass fornisce una versione migliorata del client IPC, denominata client IPC V2. Client IPC V2:

- Riduce la quantità di codice da scrivere per utilizzare le operazioni IPC e aiuta a evitare gli errori comuni che possono verificarsi con il client IPC V1.
- Richiama i callback del gestore delle sottoscrizioni in un thread separato, quindi ora è possibile eseguire codice di blocco, incluse chiamate di funzioni IPC aggiuntive, nei callback del gestore delle sottoscrizioni. Il client IPC V1 utilizza lo stesso thread per comunicare con il server IPC e chiamare i callback del gestore delle sottoscrizioni.
- Consente di chiamare le operazioni di sottoscrizione utilizzando espressioni Lambda (Java) o funzioni (Python). Il client IPC V1 richiede la definizione di classi di gestione delle sottoscrizioni.
- Fornisce versioni sincrone e asincrone di ogni operazione IPC. Il client IPC V1 fornisce solo versioni asincrone di ogni operazione.

Si consiglia di utilizzare il client IPC V2 per sfruttare questi miglioramenti. Tuttavia, molti esempi in questa documentazione e in alcuni contenuti online dimostrano solo come utilizzare il client IPC V1. È possibile utilizzare i seguenti esempi e tutorial per visualizzare componenti di esempio che utilizzano il client IPC V2:

- PublishToTopicesempi
- SubscribeToTopicesempi
- Tutorial: Sviluppa un componente Greengrass che rinvii gli aggiornamenti dei componenti
- Tutorial: Interagisci con i dispositivi IoT locali tramite MQTT

Attualmente, SDK per dispositivi AWS IoT for C++ v2 supporta solo il client IPC V1.

SDK supportati per la comunicazione tra processi

Le librerie AWS IoT Greengrass Core IPC sono incluse nelle seguenti versioni. SDK per dispositivi AWS IoT

SDK	Versione minima	Utilizzo
SDK per dispositivi AWS IoTper Java v2	v1.6.0	Per informazioni, consultare Utilizzare SDK per dispositivi AWS IoT per Java v2 (client IPC V2).
SDK per dispositivi AWS IoTper Python v2	v1.9.0	Per informazioni, consultare Uso SDK per dispositivi AWS IoT per Python v2 (client IPC V2).
SDK per dispositivi AWS IoTper C++ v2	v1.17.0	Per informazioni, consultare Utilizzare SDK per dispositivi AWS IoT per C++ v2.
SDK per dispositivi AWS IoTper v2 JavaScript	v1.12.0	Per informazioni, consultare Usa SDK per dispositivi AWS IoT per JavaScript v2 (client IPC V1).

Connect al servizio AWS IoT Greengrass Core IPC

Per utilizzare la comunicazione tra processi nel componente personalizzato, è necessario creare una connessione a un socket del server IPC eseguito dal software AWS IoT Greengrass Core. Completate le seguenti attività per scaricarlo e utilizzarlo SDK per dispositivi AWS IoT nella lingua di vostra scelta.

Utilizzare SDK per dispositivi AWS IoT per Java v2 (client IPC V2)

Per utilizzare il SDK per dispositivi AWS IoT per Java v2 (client IPC V2)

1. Scaricate il file SDK per dispositivi AWS IoTper Java v2 (v1.6.0 o successivo).
2. Effettuate una delle seguenti operazioni per eseguire il codice personalizzato nel componente:
   - Crea il tuo componente come file JAR che include ed esegui SDK per dispositivi AWS IoT questo file JAR nella ricetta del componente.
   - Definisci il SDK per dispositivi AWS IoT JAR come elemento componente e aggiungi quell'artefatto al classpath quando esegui l'applicazione nella ricetta del componente.
3. Utilizzate il codice seguente per creare il client IPC.

```
try (GreengrassCoreIPCClientV2 ipcClient =
 GreengrassCoreIPCClientV2.builder().build()) {
    // Use client.
} catch (Exception e) {
    LOGGER.log(Level.SEVERE, "Exception occurred when using IPC.", e);
    System.exit(1);
}
```

Uso SDK per dispositivi AWS IoT per Python v2 (client IPC V2)

Per usare SDK per dispositivi AWS IoT for Python v2 (client IPC V2)

1. Scarica il SDK per dispositivi AWS IoTper Python (v1.9.0 o successivo).
2. Aggiungi i passaggi di installazione dell'SDK al ciclo di vita dell'installazione nella ricetta del componente.
3. Crea una connessione al servizio AWS IoT Greengrass Core IPC. Utilizzate il codice seguente per creare il client IPC.

```
from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2

try:
    ipc_client = GreengrassCoreIPCClientV2()
    # Use IPC client.
except Exception:
    print('Exception occurred when using IPC.', file=sys.stderr)
```

```
    traceback.print_exc()
    exit(1)
```

Utilizzare SDK per dispositivi AWS IoT per C++ v2

Per creare la SDK per dispositivi AWS IoT v2 per C++, un dispositivo deve disporre dei seguenti strumenti:

- C++ 11 o versione successiva
- CMake 3.1 o successivo
- Uno dei seguenti compilatori:
  - GCC 4.8 o successivo
  - Clang 3.9 o successivo
  - MSVC 2015 o versione successiva

Per usare il per C++ SDK per dispositivi AWS IoT v2

1. Scarica la versione SDK per dispositivi AWS IoTper C++ v2 (v1.17.0 o successiva).
2. Segui le istruzioni di installazione nel README per creare il file per C++ v2 dal codice sorgenteSDK per dispositivi AWS IoT.
3. Nel tuo strumento di compilazione C++, collega la libreria IPC Greengrass`AWS::GreengrassIpc-cpp`, che hai creato nel passaggio precedente. L'`CMakeLists.txt`esempio seguente collega la libreria IPC Greengrass a un progetto creato con CMake.

   ```
   cmake_minimum_required(VERSION 3.1)
   project (greengrassv2_pubsub_subscriber)

   file(GLOB MAIN_SRC
           "*.h"
           "*.cpp"
           )
   add_executable(${PROJECT_NAME} ${MAIN_SRC})

   set_target_properties(${PROJECT_NAME} PROPERTIES
           LINKER_LANGUAGE CXX
           CXX_STANDARD 11)
   ```

```
find_package(aws-crt-cpp PATHS ~/sdk-cpp-workspace/build)
find_package(EventstreamRpc-cpp PATHS ~/sdk-cpp-workspace/build)
find_package(GreengrassIpc-cpp PATHS ~/sdk-cpp-workspace/build)
target_link_libraries(${PROJECT_NAME} AWS::GreengrassIpc-cpp)
```

4. Nel codice del componente, create una connessione al servizio AWS IoT Greengrass Core IPC per creare un client IPC (). `Aws::Greengrass::GreengrassCoreIpcClient` È necessario definire un gestore del ciclo di vita della connessione IPC che gestisca gli eventi di connessione, disconnessione ed errore IPC. L'esempio seguente crea un client IPC e un gestore del ciclo di vita della connessione IPC che stampa quando il client IPC si connette, si disconnette e rileva errori.

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        std::cout << "OnConnectCallback" << std::endl;
    }

    void OnDisconnectCallback(RpcError error) override {
        std::cout << "OnDisconnectCallback: " << error.StatusToString() <<
 std::endl;
        exit(-1);
    }

    bool OnErrorCallback(RpcError error) override {
        std::cout << "OnErrorCallback: " << error.StatusToString() << std::endl;
        return true;
    }
};

int main() {
    // Create the IPC client.
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
```

```
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    // Use the IPC client to create an operation request.

    // Activate the operation request.
    auto activate = operation.Activate(request, nullptr);
    activate.wait();

    // Wait for Greengrass Core to respond to the request.
    auto responseFuture = operation.GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from
Greengrass Core." << std::endl;
        exit(-1);
    }

    // Check the result of the request.
    auto response = responseFuture.get();
    if (response) {
        std::cout << "Successfully published to topic: " << topic << std::endl;
    } else {
        // An error occurred.
        std::cout << "Failed to publish to topic: " << topic << std::endl;
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            std::cout << "Operation error: " << error->GetMessage().value() <<
std::endl;
        } else {
            std::cout << "RPC error: " << response.GetRpcError() << std::endl;
        }
        exit(-1);
    }

    return 0;
```

```
}
```

5. Per eseguire il codice personalizzato nel componente, crea il codice come artefatto binario ed esegui l'artefatto binario nella ricetta del componente. Imposta l'`Execute`autorizzazione dell'artefatto per consentire al software AWS IoT Greengrass Core di `OWNER` eseguire l'artefatto binario.

   La `Manifests` sezione relativa alla ricetta del componente potrebbe essere simile all'esempio seguente.

   JSON

   ```
   {
     ...
     "Manifests": [
       {
         "Lifecycle": {
           "run": "{artifacts:path}/greengrassv2_pubsub_subscriber"
         },
         "Artifacts": [
           {
             "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
   com.example.PubSubSubscriberCpp/1.0.0/greengrassv2_pubsub_subscriber",
             "Permission": {
               "Execute": "OWNER"
             }
           }
         ]
       }
     ]
   }
   ```

   YAML

   ```
   ...
   Manifests:
     - Lifecycle:
         run: {artifacts:path}/greengrassv2_pubsub_subscriber
       Artifacts:
         - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
   com.example.PubSubSubscriberCpp/1.0.0/greengrassv2_pubsub_subscriber
           Permission:
   ```

```
            Execute: OWNER
```

Usa SDK per dispositivi AWS IoT per JavaScript v2 (client IPC V1)

Per creare SDK per dispositivi AWS IoT for JavaScript v2 da utilizzare con NodeJS, un dispositivo deve disporre dei seguenti strumenti:

- NodeJS 10.0 o versione successiva
  - Esegui `node -v` per controllare la versione di Node.
- CMake 3.1 o successivo

Per utilizzare SDK per dispositivi AWS IoT for JavaScript v2 (client IPC V1)

1. Scarica la versione SDK per dispositivi AWS IoTper JavaScript v2 (v1.12.10 o successiva).
2. Segui le istruzioni di installazione nel README per compilare la versione v2 dal SDK per dispositivi AWS IoT codice sorgente. JavaScript
3. Crea una connessione al servizio AWS IoT Greengrass Core IPC. Completa i seguenti passaggi per creare il client IPC e stabilire una connessione.
4. Utilizzate il codice seguente per creare il client IPC.

   ```
   import * as greengrascoreipc from 'aws-iot-device-sdk-v2';

   let client = greengrascoreipc.createClient();
   ```

5. Usa il codice seguente per stabilire una connessione dal tuo componente al nucleo Greengrass.

   ```
   await client.connect();
   ```

Autorizza i componenti a eseguire operazioni IPC

Per consentire ai componenti personalizzati di utilizzare alcune operazioni IPC, è necessario definire politiche di autorizzazione che consentano al componente di eseguire l'operazione su determinate risorse. Ogni politica di autorizzazione definisce un elenco di operazioni e un elenco di risorse consentite dalla politica. Ad esempio, il servizio IPC di messaggistica di pubblicazione/sottoscrizione definisce le operazioni di pubblicazione e sottoscrizione per le risorse tematiche. È possibile utilizzare il carattere * jolly per consentire l'accesso a tutte le operazioni o a tutte le risorse.

Le politiche di autorizzazione vengono definite con il parametro di `accessControl` configurazione, che è possibile impostare nella ricetta del componente o quando si distribuisce il componente. L'`accessControl`oggetto mappa gli identificatori del servizio IPC su elenchi di politiche di autorizzazione. È possibile definire più politiche di autorizzazione per ogni servizio IPC per controllare l'accesso. Ogni politica di autorizzazione ha un ID di policy, che deve essere univoco tra tutti i componenti.

> **Tip**
>
> Per creare ID di policy univoci, è possibile combinare il nome del componente, il nome del servizio IPC e un contatore. Ad esempio, un componente denominato `com.example.HelloWorld` potrebbe definire due politiche di autorizzazione alla pubblicazione/sottoscrizione con i seguenti ID:
>
> - `com.example.HelloWorld:pubsub:1`
> - `com.example.HelloWorld:pubsub:2`

Le politiche di autorizzazione utilizzano il seguente formato. Questo oggetto è il parametro `accessControl` di configurazione.

JSON

```
{
  "IPC service identifier": {
    "policyId": {
      "policyDescription": "description",
      "operations": [
        "operation1",
        "operation2"
      ],
      "resources": [
        "resource1",
        "resource2"
      ]
    }
  }
}
```

YAML

```
IPC service identifier:
  policyId:
    policyDescription: description
    operations:
      - operation1
      - operation2
    resources:
      - resource1
      - resource2
```

Wildcard nelle politiche di autorizzazione

È possibile utilizzare il carattere * jolly nell'`resource`elemento delle politiche di autorizzazione IPC per consentire l'accesso a più risorse in un'unica politica di autorizzazione.

- In tutte le versioni del nucleo Greengrass, è possibile specificare un singolo * carattere come risorsa per consentire l'accesso a tutte le risorse.
- In Greengrass nucleus v2.6.0 e versioni successive, puoi specificare il carattere in una risorsa in modo che corrisponda a qualsiasi combinazione di * caratteri. Ad esempio, è possibile specificare di consentire l'accesso `factory/1/devices/Thermostat*/status` a un argomento di stato per tutti i dispositivi termostati di una fabbrica, dove il nome di ogni dispositivo inizia con. `Thermostat`

Quando si definiscono le politiche di autorizzazione per il servizio AWS IoT Core MQTT IPC, è anche possibile utilizzare i caratteri jolly MQTT (+and#) per abbinare più risorse. Per ulteriori informazioni, vedete i caratteri jolly MQTT nelle politiche di autorizzazione MQTT IPC. AWS IoT Core

Variabili di ricetta nelle politiche di autorizzazione

Se si utilizza Greengrass nucleus v2.6.0 o versione successiva e si imposta l'opzione di interpolateComponentConfigurationconfigurazione di Greengrass nucleus su`true`, è possibile utilizzare la variabile recipe nelle politiche di autorizzazione. `{iot:thingName}` Quando è necessaria una politica di autorizzazione che includa il nome del dispositivo principale, ad esempio per gli argomenti MQTT o le ombre dei dispositivi, è possibile utilizzare questa variabile recipe per configurare un'unica politica di autorizzazione per un gruppo di dispositivi principali. Ad esempio, è possibile consentire a un componente l'accesso alla seguente risorsa per le operazioni IPC shadow.

```
$aws/things/{iot:thingName}/shadow/
```

Caratteri speciali nelle politiche di autorizzazione

Per specificare un valore letterale * o un ? carattere in una politica di autorizzazione, è necessario utilizzare una sequenza di escape. Le seguenti sequenze di escape indicano al software AWS IoT Greengrass Core di utilizzare il valore letterale anziché il significato speciale del carattere. Ad esempio, il * carattere è un jolly che corrisponde a qualsiasi combinazione di caratteri.

Carattere letterale	Sequenza di escape	Note
*	${*}	
?	${?}	AWS IoT Greengrassattualmente non supporta il carattere ? jolly, che corrisponde a qualsiasi carattere singolo.
$	${$}	Usa questa sequenza di escape per abbinare una risorsa che contiene${. Ad esempio, per abbinare una risorsa denominata a${resourceName} , è necessario specificare${$}{resourceName} . Altrimenti, per far corrispondere una risorsa che contiene$, è possibile utilizzare un valore letterale$, ad esempio per consentire l'accesso a un argomento che inizia con$aws.

Esempi di politiche di autorizzazione

Puoi fare riferimento ai seguenti esempi di politiche di autorizzazione per aiutarti a configurare le politiche di autorizzazione per i tuoi componenti.

Example Esempio di ricetta dei componenti con una politica di autorizzazione

Il seguente esempio di ricetta del componente include un `accessControl` oggetto che definisce una politica di autorizzazione. Questa politica autorizza il `com.example.HelloWorld` componente a pubblicare sull'`test/topic`argomento.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.HelloWorld",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that publishes messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.HelloWorld:pubsub:1": {
            "policyDescription": "Allows access to publish to test/topic.",
            "operations": [
              "aws.greengrass#PublishToTopic"
            ],
            "resources": [
              "test/topic"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "java -jar {artifacts:path}/HelloWorld.jar"
      }
    }
  ]
```

```
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.HelloWorld
ComponentVersion: '1.0.0'
ComponentDescription: A component that publishes messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        "com.example.HelloWorld:pubsub:1":
          policyDescription: Allows access to publish to test/topic.
          operations:
            - "aws.greengrass#PublishToTopic"
          resources:
            - "test/topic"
Manifests:
  - Lifecycle:
      run: |-
        java -jar {artifacts:path}/HelloWorld.jar
```

Example Esempio di aggiornamento della configurazione del componente con una politica di autorizzazione

L'esempio seguente di aggiornamento della configurazione in una distribuzione specifica di configurare un componente con un `accessControl` oggetto che definisce una politica di autorizzazione. Questa politica autorizza il `com.example.HelloWorld` componente a pubblicare sull'`test/topic`argomento.

Console

Configurazione da unire

```
{
  "accessControl": {
    "aws.greengrass.ipc.pubsub": {
      "com.example.HelloWorld:pubsub:1": {
```

```
        "policyDescription": "Allows access to publish to test/topic.",
        "operations": [
          "aws.greengrass#PublishToTopic"
        ],
        "resources": [
          "test/topic"
        ]
      }
    }
  }
}
```

AWS CLI

Il comando seguente crea una distribuzione su un dispositivo principale.

```
aws greengrassv2 create-deployment --cli-input-json file://hello-world-
deployment.json
```

Il `hello-world-deployment.json` file contiene il seguente documento JSON.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.HelloWorld": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"accessControl\":{\"aws.greengrass.ipc.pubsub\":
{\"com.example.HelloWorld:pubsub:1\":{\"policyDescription\":\"Allows access to
 publish to test/topic.\",\"operations\":[\"aws.greengrass#PublishToTopic\"],
\"resources\":[\"test/topic\"]}}}}"
      }
    }
  }
}
```

Greengrass CLI

Il seguente comando Greengrass CLI crea una distribuzione locale su un dispositivo principale.

```
sudo greengrass-cli deployment create \
```

```
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.HelloWorld=1.0.0" \
  --update-config hello-world-configuration.json
```

Il `hello-world-configuration.json` file contiene il seguente documento JSON.

```
{
  "com.example.HelloWorld": {
    "MERGE": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.HelloWorld:pubsub:1": {
            "policyDescription": "Allows access to publish to test/topic.",
            "operations": [
              "aws.greengrass#PublishToTopic"
            ],
            "resources": [
              "test/topic"
            ]
          }
        }
      }
    }
  }
}
```

Iscriviti ai flussi di eventi IPC

È possibile utilizzare le operazioni IPC per sottoscrivere flussi di eventi su un dispositivo core Greengrass. Per utilizzare un'operazione di sottoscrizione, definisci un gestore di sottoscrizioni e crea una richiesta al servizio IPC. Quindi, il client IPC esegue le funzioni del gestore delle sottoscrizioni ogni volta che il dispositivo principale trasmette un messaggio di evento al componente.

È possibile chiudere un abbonamento per interrompere l'elaborazione dei messaggi relativi agli eventi. A tale scopo, chiamate `closeStream()` (Java), `close()` (Python) o `Close()` (C++) sull'oggetto dell'operazione di sottoscrizione utilizzato per aprire l'abbonamento.

Il servizio AWS IoT Greengrass Core IPC supporta le seguenti operazioni di sottoscrizione:

- SubscribeToTopic

- SubscribeToIoTCore
- SubscribeToComponentUpdates
- SubscribeToConfigurationUpdate
- SubscribeToValidateConfigurationUpdates

Argomenti

Definire i gestori di sottoscrizione

Per definire un gestore di sottoscrizioni, definite le funzioni di callback che gestiscono i messaggi di evento, gli errori e la chiusura dei flussi. Se si utilizza il client IPC V1, è necessario definire queste funzioni in una classe. Se si utilizza il client IPC V2, disponibile nelle versioni successive degli SDK Java e Python, è possibile definire queste funzioni senza creare una classe di gestione delle sottoscrizioni.

Java

Se si utilizza il client IPC V1, è necessario implementare l'interfaccia generica. `software.amazon.awssdk.eventstreamrpc.StreamResponseHandler<`*`StreamEventType`*`>` *`StreamEventType`*è il tipo di messaggio di evento per l'operazione di sottoscrizione. Definite le seguenti funzioni per gestire i messaggi di evento, gli errori e la chiusura dei flussi.

Se si utilizza il client IPC V2, è possibile definire queste funzioni al di fuori di una classe di gestione delle sottoscrizioni o utilizzare espressioni lambda.

```
void onStreamEvent(StreamEventType event)
```

Il callback che il client IPC chiama quando riceve un messaggio di evento, ad esempio un messaggio MQTT o una notifica di aggiornamento del componente.

```
boolean onStreamError(Throwable error)
```

Il callback che il client IPC chiama quando si verifica un errore di stream.

Restituisci true per chiudere il flusso di sottoscrizioni a causa dell'errore o restituisci false per mantenere aperto lo stream.

```
void onStreamClosed()
```

Il callback che il client IPC chiama alla chiusura dello stream.

Python

Se si utilizza il client IPC V1, è necessario estendere la classe stream response handler che corrisponde all'operazione di sottoscrizione. SDK per dispositivi AWS IoTInclude una classe di gestione delle sottoscrizioni per ogni operazione di sottoscrizione. *StreamEventType*è il tipo di messaggio di evento per l'operazione di sottoscrizione. Definite le seguenti funzioni per gestire i messaggi di evento, gli errori e la chiusura dei flussi.

Se si utilizza il client IPC V2, è possibile definire queste funzioni al di fuori di una classe di gestione delle sottoscrizioni o utilizzare espressioni lambda.

```
def on_stream_event(self, event: StreamEventType) -> None
```

Il callback che il client IPC chiama quando riceve un messaggio di evento, ad esempio un messaggio MQTT o una notifica di aggiornamento del componente.

```
def on_stream_error(self, error: Exception) -> bool
```

Il callback che il client IPC chiama quando si verifica un errore di stream.

Restituisci true per chiudere il flusso di sottoscrizioni a causa dell'errore o restituisci false per mantenere aperto lo stream.

```
def on_stream_closed(self) -> None
```

Il callback che il client IPC chiama alla chiusura dello stream.

C++

Implementa una classe che deriva dalla classe stream response handler che corrisponde all'operazione di sottoscrizione. SDK per dispositivi AWS IoTInclude una classe base di gestione delle sottoscrizioni per ogni operazione di sottoscrizione. *StreamEventType*è il tipo di messaggio di evento per l'operazione di sottoscrizione. Definite le seguenti funzioni per gestire i messaggi di evento, gli errori e la chiusura dei flussi.

`void OnStreamEvent(StreamEventType *event)`

Il callback che il client IPC chiama quando riceve un messaggio di evento, ad esempio un messaggio MQTT o una notifica di aggiornamento del componente.

`bool OnStreamError(OperationError *error)`

Il callback che il client IPC chiama quando si verifica un errore di stream.

Restituisci true per chiudere il flusso di sottoscrizioni a causa dell'errore o restituisci false per mantenere aperto lo stream.

`void OnStreamClosed()`

Il callback che il client IPC chiama alla chiusura dello stream.

JavaScript

Implementa una classe che deriva dalla classe stream response handler che corrisponde all'operazione di sottoscrizione. SDK per dispositivi AWS IoTInclude una classe base di gestione delle sottoscrizioni per ogni operazione di sottoscrizione. *StreamEventType*è il tipo di messaggio di evento per l'operazione di sottoscrizione. Definite le seguenti funzioni per gestire i messaggi di evento, gli errori e la chiusura dei flussi.

`on(event: 'ended', listener: StreamingOperationEndedListener)`

Il callback che il client IPC chiama alla chiusura dello stream.

`on(event: 'streamError', listener: StreamingRpcErrorListener)`

Il callback che il client IPC chiama quando si verifica un errore di streaming.

Restituisci true per chiudere il flusso di sottoscrizioni a causa dell'errore o restituisci false per mantenere aperto lo stream.

`on(event: 'message', listener: (message: InboundMessageType) => void)`

Il callback che il client IPC chiama quando riceve un messaggio di evento, ad esempio un messaggio MQTT o una notifica di aggiornamento del componente.

Esempi di gestori di sottoscrizioni

L'esempio seguente mostra come utilizzare l'SubscribeToTopicoperazione e un gestore di sottoscrizioni per sottoscrivere i messaggi di pubblicazione/sottoscrizione locali.

Java (IPC client V2)

Example Esempio: iscriversi ai messaggi locali di pubblicazione/sottoscrizione

```
package com.aws.greengrass.docs.samples.ipc;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClientV2;
import software.amazon.awssdk.aws.greengrass.SubscribeToTopicResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.*;

import java.nio.charset.StandardCharsets;
import java.util.Optional;

public class SubscribeToTopicV2 {

    public static void main(String[] args) {
        String topic = args[0];
        try (GreengrassCoreIPCClientV2 ipcClient =
 GreengrassCoreIPCClientV2.builder().build()) {
            SubscribeToTopicRequest request = new
 SubscribeToTopicRequest().withTopic(topic);
            GreengrassCoreIPCClientV2.StreamingResponse<SubscribeToTopicResponse,
                    SubscribeToTopicResponseHandler> response =
                    ipcClient.subscribeToTopic(request,
 SubscribeToTopicV2::onStreamEvent,
                            Optional.of(SubscribeToTopicV2::onStreamError),
                            Optional.of(SubscribeToTopicV2::onStreamClosed));
            SubscribeToTopicResponseHandler responseHandler =
 response.getHandler();
            System.out.println("Successfully subscribed to topic: " + topic);

            // Keep the main thread alive, or the process will exit.
            try {
                while (true) {
                    Thread.sleep(10000);
                }
            } catch (InterruptedException e) {
                System.out.println("Subscribe interrupted.");
            }

            // To stop subscribing, close the stream.
            responseHandler.closeStream();
        } catch (Exception e) {
            if (e.getCause() instanceof UnauthorizedError) {
```

```
                System.err.println("Unauthorized error while publishing to topic: "
 + topic);
            } else {
                System.err.println("Exception occurred when using IPC.");
            }
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static void onStreamEvent(SubscriptionResponseMessage
 subscriptionResponseMessage) {
        try {
            BinaryMessage binaryMessage =
 subscriptionResponseMessage.getBinaryMessage();
            String message = new String(binaryMessage.getMessage(),
 StandardCharsets.UTF_8);
            String topic = binaryMessage.getContext().getTopic();
            System.out.printf("Received new message on topic %s: %s%n", topic,
 message);
        } catch (Exception e) {
            System.err.println("Exception occurred while processing subscription
 response " +
                    "message.");
            e.printStackTrace();
        }
    }

    public static boolean onStreamError(Throwable error) {
        System.err.println("Received a stream error.");
        error.printStackTrace();
        return false; // Return true to close stream, false to keep stream open.
    }

    public static void onStreamClosed() {
        System.out.println("Subscribe to topic stream closed.");
    }
}
```

Python (IPC client V2)

Example Esempio: iscriversi ai messaggi locali di pubblicazione/sottoscrizione

```
import sys
```

```
import time
import traceback

from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2
from awsiot.greengrasscoreipc.model import (
    SubscriptionResponseMessage,
    UnauthorizedError
)


def main():
    args = sys.argv[1:]
    topic = args[0]

    try:
        ipc_client = GreengrassCoreIPCClientV2()
        # Subscription operations return a tuple with the response and the
 operation.
        _, operation = ipc_client.subscribe_to_topic(topic=topic,
 on_stream_event=on_stream_event,

 on_stream_error=on_stream_error, on_stream_closed=on_stream_closed)
        print('Successfully subscribed to topic: ' + topic)

        # Keep the main thread alive, or the process will exit.
        try:
            while True:
                time.sleep(10)
        except InterruptedError:
            print('Subscribe interrupted.')

        # To stop subscribing, close the stream.
        operation.close()
    except UnauthorizedError:
        print('Unauthorized error while subscribing to topic: ' +
              topic, file=sys.stderr)
        traceback.print_exc()
        exit(1)
    except Exception:
        print('Exception occurred', file=sys.stderr)
        traceback.print_exc()
        exit(1)
```

```
def on_stream_event(event: SubscriptionResponseMessage) -> None:
    try:
        message = str(event.binary_message.message, 'utf-8')
        topic = event.binary_message.context.topic
        print('Received new message on topic %s: %s' % (topic, message))
    except:
        traceback.print_exc()


def on_stream_error(error: Exception) -> bool:
    print('Received a stream error.', file=sys.stderr)
    traceback.print_exc()
    return False  # Return True to close stream, False to keep stream open.


def on_stream_closed() -> None:
    print('Subscribe to topic stream closed.')


if __name__ == '__main__':
    main()
```

C++

Example Esempio: iscriversi ai messaggi locali di pubblicazione/sottoscrizione

```
#include <iostream>

#include </crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class SubscribeResponseHandler : public SubscribeToTopicStreamHandler {
    public:
        virtual ~SubscribeResponseHandler() {}

    private:
        void OnStreamEvent(SubscriptionResponseMessage *response) override {
            auto jsonMessage = response->GetJsonMessage();
            if (jsonMessage.has_value() &&
 jsonMessage.value().GetMessage().has_value()) {
```

```
                auto messageString =
 jsonMessage.value().GetMessage().value().View().WriteReadable();
                // Handle JSON message.
            } else {
                auto binaryMessage = response->GetBinaryMessage();
                if (binaryMessage.has_value() &&
 binaryMessage.value().GetMessage().has_value()) {
                    auto messageBytes = binaryMessage.value().GetMessage().value();
                    std::string messageString(messageBytes.begin(),
 messageBytes.end());
                    // Handle binary message.
                }
            }
        }

        bool OnStreamError(OperationError *error) override {
            // Handle error.
            return false; // Return true to close stream, false to keep stream open.
        }

        void OnStreamClosed() override {
            // Handle close.
        }
};

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        // Handle connection to IPC service.
    }

    void OnDisconnectCallback(RpcError error) override {
        // Handle disconnection from IPC service.
    }

    bool OnErrorCallback(RpcError error) override {
        // Handle IPC service connection error.
        return true;
    }
};

int main() {
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
```

```
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    String topic("my/topic");
    int timeout = 10;

    SubscribeToTopicRequest request;
    request.SetTopic(topic);

    //SubscribeResponseHandler streamHandler;
    auto streamHandler = MakeShared<SubscribeResponseHandler>(DefaultAllocator());
    auto operation = ipcClient.NewSubscribeToTopic(streamHandler);
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (!response) {
        // Handle error.
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            (void)error;
            // Handle operation error.
        } else {
            // Handle RPC error.
        }
        exit(-1);
    }
```

```
    // Keep the main thread alive, or the process will exit.
    while (true) {
        std::this_thread::sleep_for(std::chrono::seconds(10));
    }

    operation->Close();
    return 0;
}
```

JavaScript

Example Esempio: iscriversi ai messaggi locali di pubblicazione/sottoscrizione

```
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";
import {SubscribeToTopicRequest, SubscriptionResponseMessage} from "aws-iot-device-
sdk-v2/dist/greengrasscoreipc/model";
import {RpcError} from "aws-iot-device-sdk-v2/dist/eventstream_rpc";

class SubscribeToTopic {
    private ipcClient : greengrasscoreipc.Client
    private readonly topic : string;

    constructor() {
        // define your own constructor, e.g.
        this.topic = "<define_your_topic>";
        this.subscribeToTopic().then(r => console.log("Started workflow"));
    }

    private async subscribeToTopic() {
        try {
            this.ipcClient = await getIpcClient();

            const subscribeToTopicRequest : SubscribeToTopicRequest = {
                topic: this.topic,
            }

            const streamingOperation =
 this.ipcClient.subscribeToTopic(subscribeToTopicRequest, undefined); //
 conditionally apply options

            streamingOperation.on("message", (message: SubscriptionResponseMessage)
 => {
                // parse the message depending on your use cases, e.g.
```

```
                if(message.binaryMessage && message.binaryMessage.message) {
                    const receivedMessage =
 message.binaryMessage?.message.toString();
                }
            });

            streamingOperation.on("streamError", (error : RpcError) => {
                // define your own error handling logic
            })

            streamingOperation.on("ended", () => {
                // define your own logic
            })

            await streamingOperation.activate();

            // Keep the main thread alive, or the process will exit.
            await new Promise((resolve) => setTimeout(resolve, 10000))
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}

export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

// starting point
const subscribeToTopic = new SubscribeToTopic();
```

Migliori pratiche IPC

Le migliori pratiche per l'utilizzo di IPC nei componenti personalizzati differiscono tra il client IPC V1 e il client IPC V2. Segui le migliori pratiche per la versione del client IPC che utilizzi.

IPC client V2

Il client IPC V2 esegue le funzioni di callback in un thread separato, quindi rispetto al client IPC V1, ci sono meno linee guida da seguire quando si utilizzano IPC e si scrivono funzioni di gestione delle sottoscrizioni.

- Riutilizza un client IPC

  Dopo aver creato un client IPC, tienilo aperto e riutilizzalo per tutte le operazioni IPC. La creazione di più client utilizza risorse aggiuntive e può causare perdite di risorse.

- Gestisci le eccezioni

  Il client IPC V2 registra le eccezioni non rilevate nelle funzioni di gestione delle sottoscrizioni. È necessario catturare le eccezioni nelle funzioni del gestore per gestire gli errori che si verificano nel codice.

IPC client V1

Il client IPC V1 utilizza un singolo thread che comunica con il server IPC e chiama i gestori di sottoscrizione. È necessario considerare questo comportamento sincrono quando si scrivono funzioni di gestione delle sottoscrizioni.

- Riutilizza un client IPC

  Dopo aver creato un client IPC, tienilo aperto e riutilizzalo per tutte le operazioni IPC. La creazione di più client utilizza risorse aggiuntive e può causare perdite di risorse.

- Esegui il codice di blocco in modo asincrono

  Il client IPC V1 non può inviare nuove richieste o elaborare nuovi messaggi di eventi mentre il thread è bloccato. È necessario eseguire il codice di blocco in un thread separato eseguito dalla funzione di gestione. Il codice di blocco include `sleep` chiamate, loop in esecuzione continua e richieste di I/O sincrone il cui completamento richiede tempo.

- Invia nuove richieste IPC in modo asincrono

Il client IPC V1 non può inviare una nuova richiesta dall'interno delle funzioni di gestione delle sottoscrizioni, poiché la richiesta blocca la funzione di gestione se si attende una risposta. È necessario inviare le richieste IPC in un thread separato eseguito dalla funzione di gestione.

- Gestisci le eccezioni

  Il client IPC V1 non gestisce le eccezioni non rilevate nelle funzioni di gestione delle sottoscrizioni. Se la funzione di gestione genera un'eccezione, l'abbonamento si chiude e l'eccezione non viene visualizzata nei registri dei componenti. È necessario catturare le eccezioni nelle funzioni del gestore per mantenere aperto l'abbonamento e registrare gli errori che si verificano nel codice.

Pubblicare/sottoscrivere messaggi locali

La messaggistica Publish/subscribe (pubsub) consente di inviare e ricevere messaggi relativi agli argomenti. I componenti possono pubblicare messaggi su argomenti per inviare messaggi ad altri componenti. Quindi, i componenti sottoscritti a quell'argomento possono agire sui messaggi che ricevono.

> **Note**
>
> Non è possibile utilizzare questo servizio IPC di pubblicazione/sottoscrizione per pubblicare o sottoscrivere MQTT. AWS IoT Core Per ulteriori informazioni su come scambiare messaggi con MQTT, vedere. AWS IoT Core AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

Argomenti

Versioni SDK minime

La tabella seguente elenca le versioni minime da utilizzare per pubblicare e sottoscrivere messaggi da e verso argomenti locali. SDK per dispositivi AWS IoT

SDK	Versione minima	
SDK per dispositivi AWS IoTper Java v2	v1.2.10	
SDK per dispositivi AWS IoTper Python v2	v1.5.3	
SDK per dispositivi AWS IoTper C++ v2	v1.17.0	
SDK per dispositivi AWS IoT per JavaScript v2	v1.12.0	

Autorizzazione

Per utilizzare la messaggistica locale di pubblicazione/sottoscrizione in un componente personalizzato, è necessario definire politiche di autorizzazione che consentano al componente di inviare e ricevere messaggi sugli argomenti. Per informazioni sulla definizione delle politiche di autorizzazione, vedere. Autorizza i componenti a eseguire operazioni IPC

Le politiche di autorizzazione per i messaggi di pubblicazione/sottoscrizione hanno le seguenti proprietà.

Identificatore del servizio IPC: `aws.greengrass.ipc.pubsub`

Operazione	Descrizione	Risorse
`aws.greengrass#PublishToTopic`	Consente a un componente di pubblicare messaggi sugli argomenti specificati dall'utente.	Una stringa di argomento, ad esempio`test/topic` . Usa an `*` per abbinare qualsiasi combinazione di caratteri in un argomento.

Operazione	Descrizione	Risorse
		Questa stringa di argomento non supporta i caratteri jolly degli argomenti MQTT (#and+).
`aws.greengrass#SubscribeToTopic`	Consente a un componente di sottoscrivere i messaggi per gli argomenti specificati.	Una stringa di argomento, ad esempio`test/topic` . Usa an * per abbinare qualsiasi combinazione di caratteri in un argomento. In Greengrass nucleus v2.6.0 e versioni successive, è possibile sottoscrivere argomenti che contengono caratteri jolly degli argomenti MQTT (and). # + Questa stringa di argomenti supporta i caratteri jolly degli argomenti MQTT come caratteri letterali . Ad esempio, se la politica di autorizzazione di un componente concede l'accesso a`test/topic/#` , il componente può sottoscrivere`test/topic/#` , ma non può sottoscrivere. `test/topic/filter`

Operazione	Descrizione	Risorse
*	Consente a un componente di pubblicare e sottoscrivere messaggi per gli argomenti specificati.	Una stringa di argomento, ad esempio`test/topic` . Usa an * per abbinare qualsiasi combinazione di caratteri in un argomento. In Greengrass nucleus v2.6.0 e versioni successive, è possibile sottoscrivere argomenti che contengono caratteri jolly degli argomenti MQTT (and). # + Questa stringa di argomenti supporta i caratteri jolly degli argomenti MQTT come caratteri letterali . Ad esempio, se la politica di autorizzazione di un componente concede l'accesso a`test/topic/#` , il componente può sottoscrivere`test/topic/#` , ma non può sottoscrivere. `test/topic/filter`

Esempi di politiche di autorizzazione

È possibile fare riferimento al seguente esempio di politica di autorizzazione per configurare le politiche di autorizzazione per i componenti.

Example Esempio di politica di autorizzazione

Il seguente esempio di politica di autorizzazione consente a un componente di pubblicare e sottoscrivere tutti gli argomenti.

```
{
  "accessControl": {
```

```
    "aws.greengrass.ipc.pubsub": {
      "com.example.MyLocalPubSubComponent:pubsub:1": {
        "policyDescription": "Allows access to publish/subscribe to all topics.",
        "operations": [
          "aws.greengrass#PublishToTopic",
          "aws.greengrass#SubscribeToTopic"
        ],
        "resources": [
          "*"
        ]
      }
    }
  }
}
```

PublishToTopic

Pubblicare un messaggio in un argomento.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`topic`

L'argomento su cui pubblicare il messaggio.

`publishMessage`(Python:) `publish_message`

Il messaggio da pubblicare. Questo oggetto contiene `PublishMessage` le seguenti informazioni. È necessario specificare uno tra `jsonMessage` e`binaryMessage`.

`jsonMessage`(Python:) `json_message`

(Facoltativo) Un messaggio JSON. Questo oggetto contiene `JsonMessage` le seguenti informazioni:

`message`

Il messaggio JSON come oggetto.

`context`

Il contesto del messaggio, ad esempio l'argomento in cui è stato pubblicato il messaggio.

Questa funzionalità è disponibile per la versione 2.6.0 e successive del componente Greengrass nucleus. La tabella seguente elenca le versioni minime di da utilizzare per SDK per dispositivi AWS IoT accedere al contesto del messaggio.

SDK	Versione minima	
SDK per dispositivi AWS IoTper Java v2	v1.9.3	
SDK per dispositivi AWS IoTper Python v2	v1.11.3	
SDK per dispositivi AWS IoTper C++ v2	v1.18.4	
SDK per dispositivi AWS IoT per JavaScript v2	v1.12.0	

> **Note**
>
> Il software AWS IoT Greengrass Core utilizza gli stessi oggetti di messaggio nelle operazioni and. `PublishToTopic` `SubscribeToTopic` Il software AWS IoT Greengrass Core imposta questo oggetto contestuale nei messaggi al momento della sottoscrizione e lo ignora nei messaggi pubblicati.

Questo oggetto contiene `MessageContext` le seguenti informazioni:

`topic`

L'argomento in cui è stato pubblicato il messaggio.

`binaryMessage`(Python:) `binary_message`

(Facoltativo) Un messaggio binario. Questo oggetto contiene `BinaryMessage` le seguenti informazioni:

`message`

Il messaggio binario come blob.

`context`

Il contesto del messaggio, ad esempio l'argomento in cui è stato pubblicato il messaggio.

Questa funzionalità è disponibile per la versione 2.6.0 e successive del componente Greengrass nucleus. La tabella seguente elenca le versioni minime di da utilizzare per SDK per dispositivi AWS IoT accedere al contesto del messaggio.

SDK	Versione minima
SDK per dispositivi AWS IoTper Java v2	v1.9.3
SDK per dispositivi AWS IoTper Python v2	v1.11.3
SDK per dispositivi AWS IoTper C++ v2	v1.18.4
SDK per dispositivi AWS IoT per JavaScript v2	v1.12.0

> **Note**
>
> Il software AWS IoT Greengrass Core utilizza gli stessi oggetti di messaggio nelle operazioni and. `PublishToTopic` `SubscribeToTopic` Il software AWS IoT Greengrass Core imposta questo oggetto contestuale nei messaggi al momento della sottoscrizione e lo ignora nei messaggi pubblicati.

Questo oggetto contiene `MessageContext` le seguenti informazioni:

`topic`

L'argomento in cui è stato pubblicato il messaggio.

Risposta

Questa operazione non fornisce alcuna informazione nella sua risposta.

Esempi

Gli esempi seguenti mostrano come chiamare questa operazione nel codice componente personalizzato.

Java (IPC client V2)

Example Esempio: pubblicare un messaggio binario

```
package com.aws.greengrass.docs.samples.ipc;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClientV2;
import software.amazon.awssdk.aws.greengrass.model.BinaryMessage;
import software.amazon.awssdk.aws.greengrass.model.PublishMessage;
import software.amazon.awssdk.aws.greengrass.model.PublishToTopicRequest;
import software.amazon.awssdk.aws.greengrass.model.PublishToTopicResponse;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;

import java.nio.charset.StandardCharsets;

public class PublishToTopicV2 {

    public static void main(String[] args) {
        String topic = args[0];
        String message = args[1];
        try (GreengrassCoreIPCClientV2 ipcClient =
 GreengrassCoreIPCClientV2.builder().build()) {
            PublishToTopicV2.publishBinaryMessageToTopic(ipcClient, topic,
 message);
            System.out.println("Successfully published to topic: " + topic);
        } catch (Exception e) {
            if (e.getCause() instanceof UnauthorizedError) {
                System.err.println("Unauthorized error while publishing to topic: "
 + topic);
            } else {
                System.err.println("Exception occurred when using IPC.");
            }
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static PublishToTopicResponse publishBinaryMessageToTopic(
```

```
            GreengrassCoreIPCClientV2 ipcClient, String topic, String message)
 throws InterruptedException {
        BinaryMessage binaryMessage =
                new
 BinaryMessage().withMessage(message.getBytes(StandardCharsets.UTF_8));
        PublishMessage publishMessage = new
 PublishMessage().withBinaryMessage(binaryMessage);
        PublishToTopicRequest publishToTopicRequest =
                new
 PublishToTopicRequest().withTopic(topic).withPublishMessage(publishMessage);
        return ipcClient.publishToTopic(publishToTopicRequest);
    }
}
```

Python (IPC client V2)

Example Esempio: pubblicare un messaggio binario

```
import sys
import traceback

from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2
from awsiot.greengrasscoreipc.model import (
    PublishMessage,
    BinaryMessage
)


def main():
    args = sys.argv[1:]
    topic = args[0]
    message = args[1]

    try:
        ipc_client = GreengrassCoreIPCClientV2()
        publish_binary_message_to_topic(ipc_client, topic, message)
        print('Successfully published to topic: ' + topic)
    except Exception:
        print('Exception occurred', file=sys.stderr)
        traceback.print_exc()
        exit(1)


def publish_binary_message_to_topic(ipc_client, topic, message):
```

```
    binary_message = BinaryMessage(message=bytes(message, 'utf-8'))
    publish_message = PublishMessage(binary_message=binary_message)
    return ipc_client.publish_to_topic(topic=topic,
 publish_message=publish_message)


if __name__ == '__main__':
    main()
```

C++

Example Esempio: pubblicare un messaggio binario

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        // Handle connection to IPC service.
    }

    void OnDisconnectCallback(RpcError error) override {
        // Handle disconnection from IPC service.
    }

    bool OnErrorCallback(RpcError error) override {
        // Handle IPC service connection error.
        return true;
    }
};

int main() {
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
```

```
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    String topic("my/topic");
    String message("Hello, World!");
    int timeout = 10;

    PublishToTopicRequest request;
    Vector<uint8_t> messageData({message.begin(), message.end()});
    BinaryMessage binaryMessage;
    binaryMessage.SetMessage(messageData);
    PublishMessage publishMessage;
    publishMessage.SetBinaryMessage(binaryMessage);
    request.SetTopic(topic);
    request.SetPublishMessage(publishMessage);

    auto operation = ipcClient.NewPublishToTopic();
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (!response) {
        // Handle error.
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            (void)error;
            // Handle operation error.
        } else {
            // Handle RPC error.
        }
    }
    return 0;
```

```
}
```

JavaScript

Example Esempio: pubblicare un messaggio binario

```
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";
import {BinaryMessage, PublishMessage, PublishToTopicRequest} from "aws-iot-device-
sdk-v2/dist/greengrasscoreipc/model";

class PublishToTopic {
    private ipcClient : greengrasscoreipc.Client
    private readonly topic : string;
    private readonly messageString : string;

    constructor() {
        // define your own constructor, e.g.
        this.topic = "<define_your_topic>";
        this.messageString = "<define_your_message_string>";
        this.publishToTopic().then(r => console.log("Started workflow"));
    }

    private async publishToTopic() {
        try {
            this.ipcClient = await getIpcClient();

            const binaryMessage : BinaryMessage = {
                message: this.messageString
            }

            const publishMessage : PublishMessage = {
                binaryMessage: binaryMessage
            }

            const request : PublishToTopicRequest = {
                topic: this.topic,
                publishMessage: publishMessage
            }

            this.ipcClient.publishToTopic(request).finally(() =>
 console.log(`Published message ${publishMessage.binaryMessage?.message} to topic`))
```

```
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}


export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

// starting point
const publishToTopic = new PublishToTopic();
```

SubscribeToTopic

Iscriviti ai messaggi su un argomento.

Questa operazione è un'operazione di sottoscrizione in cui ci si iscrive a un flusso di messaggi di eventi. Per utilizzare questa operazione, definite un gestore di risposte di flusso con funzioni che gestiscono i messaggi di evento, gli errori e la chiusura dei flussi. Per ulteriori informazioni, consulta Iscriviti ai flussi di eventi IPC.

Tipo di messaggio di evento: `SubscriptionResponseMessage`

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`topic`

L'argomento a cui iscriversi.

> **Note**
>
> In Greengrass nucleus v2.6.0 e versioni successive, questo argomento supporta i caratteri jolly dell'argomento MQTT (and). # +

`receiveMode`(Python:) `receive_mode`

(Facoltativo) Il comportamento che specifica se il componente riceve messaggi da se stesso. È possibile modificare questo comportamento per consentire a un componente di agire sui propri messaggi. Il comportamento predefinito dipende dal fatto che l'argomento contenga un jolly MQTT. Seleziona una delle opzioni seguenti:

- `RECEIVE_ALL_MESSAGES`— Ricevi tutti i messaggi che corrispondono all'argomento, inclusi i messaggi del componente che effettua la sottoscrizione.

  Questa modalità è l'opzione predefinita quando ci si iscrive a un argomento che non contiene caratteri jolly MQTT.
- `RECEIVE_MESSAGES_FROM_OTHERS`— Ricevi tutti i messaggi che corrispondono all'argomento, ad eccezione dei messaggi del componente che sottoscrive.

  Questa modalità è l'opzione predefinita quando ci si iscrive a un argomento che contiene un metacarattero MQTT.

Questa funzionalità è disponibile per la versione 2.6.0 e successive del componente Greengrass nucleus. La tabella seguente elenca le versioni minime di da utilizzare per SDK per dispositivi AWS IoT impostare la modalità di ricezione.

SDK	Versione minima	
SDK per dispositivi AWS IoTper Java v2	v1.9.3	
SDK per dispositivi AWS IoTper Python v2	v1.11.3	

SDK	Versione minima	
SDK per dispositivi AWS IoTper C++ v2	v1.18.4	
SDK per dispositivi AWS IoTper v2 JavaScript	v1.12.0	

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`messages`

Il flusso di messaggi. Questo oggetto contiene `SubscriptionResponseMessage` le seguenti informazioni. Ogni messaggio contiene `jsonMessage` o `binaryMessage`.

`jsonMessage`(Python:) `json_message`

(Facoltativo) Un messaggio JSON. Questo oggetto contiene `JsonMessage` le seguenti informazioni:

`message`

Il messaggio JSON come oggetto.

`context`

Il contesto del messaggio, ad esempio l'argomento in cui è stato pubblicato il messaggio.

Questa funzionalità è disponibile per la versione 2.6.0 e successive del componente Greengrass nucleus. La tabella seguente elenca le versioni minime di da utilizzare per SDK per dispositivi AWS IoT accedere al contesto del messaggio.

SDK	Versione minima	
SDK per dispositivi AWS IoTper Java v2	v1.9.3	
SDK per dispositivi AWS IoTper Python v2	v1.11.3	

SDK	Versione minima
SDK per dispositivi AWS IoTper C++ v2	v1.18.4
SDK per dispositivi AWS IoT per JavaScript v2	v1.12.0

> **Note**
>
> Il software AWS IoT Greengrass Core utilizza gli stessi oggetti di messaggio nelle operazioni and. `PublishToTopic` `SubscribeToTopic` Il software AWS IoT Greengrass Core imposta questo oggetto contestuale nei messaggi al momento della sottoscrizione e lo ignora nei messaggi pubblicati.

Questo oggetto contiene `MessageContext` le seguenti informazioni:

`topic`

L'argomento in cui è stato pubblicato il messaggio.

`binaryMessage`(Python:) `binary_message`

(Facoltativo) Un messaggio binario. Questo oggetto contiene `BinaryMessage` le seguenti informazioni:

`message`

Il messaggio binario come blob.

`context`

Il contesto del messaggio, ad esempio l'argomento in cui è stato pubblicato il messaggio.

Questa funzionalità è disponibile per la versione 2.6.0 e successive del componente Greengrass nucleus. La tabella seguente elenca le versioni minime di da utilizzare per SDK per dispositivi AWS IoT accedere al contesto del messaggio.

SDK	Versione minima
SDK per dispositivi AWS IoTper Java v2	v1.9.3
SDK per dispositivi AWS IoTper Python v2	v1.11.3
SDK per dispositivi AWS IoTper C++ v2	v1.18.4
SDK per dispositivi AWS IoT per JavaScript v2	v1.12.0

> **Note**
>
> Il software AWS IoT Greengrass Core utilizza gli stessi oggetti di messaggio nelle operazioni and. `PublishToTopic` `SubscribeToTopic` Il software AWS IoT Greengrass Core imposta questo oggetto contestuale nei messaggi al momento della sottoscrizione e lo ignora nei messaggi pubblicati.

Questo oggetto contiene `MessageContext` le seguenti informazioni:

`topic`

L'argomento in cui è stato pubblicato il messaggio.

`topicName`(Python:) `topic_name`

L'argomento su cui è stato pubblicato il messaggio.

> **Note**
>
> Questa proprietà non è attualmente utilizzata. In Greengrass nucleus v2.6.0 e versioni successive, puoi ottenere il `(jsonMessage|binaryMessage).context.topic` valore da `SubscriptionResponseMessage` a per ottenere l'argomento in cui è stato pubblicato il messaggio.

Esempi

I seguenti esempi mostrano come chiamare questa operazione nel codice componente personalizzato.

Java (IPC client V2)

Example Esempio: iscriviti ai messaggi locali di pubblicazione/sottoscrizione

```
package com.aws.greengrass.docs.samples.ipc;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClientV2;
import software.amazon.awssdk.aws.greengrass.SubscribeToTopicResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.*;

import java.nio.charset.StandardCharsets;
import java.util.Optional;

public class SubscribeToTopicV2 {

    public static void main(String[] args) {
        String topic = args[0];
        try (GreengrassCoreIPCClientV2 ipcClient =
 GreengrassCoreIPCClientV2.builder().build()) {
            SubscribeToTopicRequest request = new
 SubscribeToTopicRequest().withTopic(topic);
            GreengrassCoreIPCClientV2.StreamingResponse<SubscribeToTopicResponse,
                    SubscribeToTopicResponseHandler> response =
                    ipcClient.subscribeToTopic(request,
 SubscribeToTopicV2::onStreamEvent,
                            Optional.of(SubscribeToTopicV2::onStreamError),
                            Optional.of(SubscribeToTopicV2::onStreamClosed));
            SubscribeToTopicResponseHandler responseHandler =
 response.getHandler();
            System.out.println("Successfully subscribed to topic: " + topic);

            // Keep the main thread alive, or the process will exit.
            try {
                while (true) {
                    Thread.sleep(10000);
                }
            } catch (InterruptedException e) {
                System.out.println("Subscribe interrupted.");
```

```
            }

            // To stop subscribing, close the stream.
            responseHandler.closeStream();
        } catch (Exception e) {
            if (e.getCause() instanceof UnauthorizedError) {
                System.err.println("Unauthorized error while publishing to topic: "
+ topic);
            } else {
                System.err.println("Exception occurred when using IPC.");
            }
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static void onStreamEvent(SubscriptionResponseMessage
subscriptionResponseMessage) {
        try {
            BinaryMessage binaryMessage =
subscriptionResponseMessage.getBinaryMessage();
            String message = new String(binaryMessage.getMessage(),
StandardCharsets.UTF_8);
            String topic = binaryMessage.getContext().getTopic();
            System.out.printf("Received new message on topic %s: %s%n", topic,
message);
        } catch (Exception e) {
            System.err.println("Exception occurred while processing subscription
response " +
                    "message.");
            e.printStackTrace();
        }
    }

    public static boolean onStreamError(Throwable error) {
        System.err.println("Received a stream error.");
        error.printStackTrace();
        return false; // Return true to close stream, false to keep stream open.
    }

    public static void onStreamClosed() {
        System.out.println("Subscribe to topic stream closed.");
    }
```

```
}
```

Python (IPC client V2)

Example Esempio: iscriversi ai messaggi locali di pubblicazione/sottoscrizione

```
import sys
import time
import traceback

from awsiot.greengrasscoreipc.clientv2 import GreengrassCoreIPCClientV2
from awsiot.greengrasscoreipc.model import (
    SubscriptionResponseMessage,
    UnauthorizedError
)


def main():
    args = sys.argv[1:]
    topic = args[0]

    try:
        ipc_client = GreengrassCoreIPCClientV2()
        # Subscription operations return a tuple with the response and the
 operation.
        _, operation = ipc_client.subscribe_to_topic(topic=topic,
 on_stream_event=on_stream_event,

 on_stream_error=on_stream_error, on_stream_closed=on_stream_closed)
        print('Successfully subscribed to topic: ' + topic)

        # Keep the main thread alive, or the process will exit.
        try:
            while True:
                time.sleep(10)
        except InterruptedError:
            print('Subscribe interrupted.')

        # To stop subscribing, close the stream.
        operation.close()
    except UnauthorizedError:
        print('Unauthorized error while subscribing to topic: ' +
              topic, file=sys.stderr)
        traceback.print_exc()
```

```
        exit(1)
    except Exception:
        print('Exception occurred', file=sys.stderr)
        traceback.print_exc()
        exit(1)


def on_stream_event(event: SubscriptionResponseMessage) -> None:
    try:
        message = str(event.binary_message.message, 'utf-8')
        topic = event.binary_message.context.topic
        print('Received new message on topic %s: %s' % (topic, message))
    except:
        traceback.print_exc()


def on_stream_error(error: Exception) -> bool:
    print('Received a stream error.', file=sys.stderr)
    traceback.print_exc()
    return False  # Return True to close stream, False to keep stream open.


def on_stream_closed() -> None:
    print('Subscribe to topic stream closed.')


if __name__ == '__main__':
    main()
```

C++

Example Esempio: iscriversi ai messaggi locali di pubblicazione/sottoscrizione

```
#include <iostream>

#include </crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class SubscribeResponseHandler : public SubscribeToTopicStreamHandler {
    public:
        virtual ~SubscribeResponseHandler() {}
```

```
    private:
        void OnStreamEvent(SubscriptionResponseMessage *response) override {
            auto jsonMessage = response->GetJsonMessage();
            if (jsonMessage.has_value() &&
 jsonMessage.value().GetMessage().has_value()) {
                auto messageString =
 jsonMessage.value().GetMessage().value().View().WriteReadable();
                // Handle JSON message.
            } else {
                auto binaryMessage = response->GetBinaryMessage();
                if (binaryMessage.has_value() &&
 binaryMessage.value().GetMessage().has_value()) {
                    auto messageBytes = binaryMessage.value().GetMessage().value();
                    std::string messageString(messageBytes.begin(),
 messageBytes.end());
                    // Handle binary message.
                }
            }
        }

        bool OnStreamError(OperationError *error) override {
            // Handle error.
            return false; // Return true to close stream, false to keep stream open.
        }

        void OnStreamClosed() override {
            // Handle close.
        }
};

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        // Handle connection to IPC service.
    }

    void OnDisconnectCallback(RpcError error) override {
        // Handle disconnection from IPC service.
    }

    bool OnErrorCallback(RpcError error) override {
        // Handle IPC service connection error.
        return true;
    }
```

```
};

int main() {
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    String topic("my/topic");
    int timeout = 10;

    SubscribeToTopicRequest request;
    request.SetTopic(topic);

    //SubscribeResponseHandler streamHandler;
    auto streamHandler = MakeShared<SubscribeResponseHandler>(DefaultAllocator());
    auto operation = ipcClient.NewSubscribeToTopic(streamHandler);
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
 Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (!response) {
        // Handle error.
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            (void)error;
            // Handle operation error.
```

```
        } else {
            // Handle RPC error.
        }
        exit(-1);
    }

    // Keep the main thread alive, or the process will exit.
    while (true) {
        std::this_thread::sleep_for(std::chrono::seconds(10));
    }

    operation->Close();
    return 0;
}
```

JavaScript

Example Esempio: iscriversi ai messaggi locali di pubblicazione/sottoscrizione

```
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";
import {SubscribeToTopicRequest, SubscriptionResponseMessage} from "aws-iot-device-
sdk-v2/dist/greengrasscoreipc/model";
import {RpcError} from "aws-iot-device-sdk-v2/dist/eventstream_rpc";

class SubscribeToTopic {
    private ipcClient : greengrasscoreipc.Client
    private readonly topic : string;

    constructor() {
        // define your own constructor, e.g.
        this.topic = "<define_your_topic>";
        this.subscribeToTopic().then(r => console.log("Started workflow"));
    }

    private async subscribeToTopic() {
        try {
            this.ipcClient = await getIpcClient();

            const subscribeToTopicRequest : SubscribeToTopicRequest = {
                topic: this.topic,
            }
```

```
            const streamingOperation =
 this.ipcClient.subscribeToTopic(subscribeToTopicRequest, undefined); //
 conditionally apply options

            streamingOperation.on("message", (message: SubscriptionResponseMessage)
 => {
                // parse the message depending on your use cases, e.g.
                if(message.binaryMessage && message.binaryMessage.message) {
                    const receivedMessage =
 message.binaryMessage?.message.toString();
                }
            });

            streamingOperation.on("streamError", (error : RpcError) => {
                // define your own error handling logic
            })

            streamingOperation.on("ended", () => {
                // define your own logic
            })

            await streamingOperation.activate();

            // Keep the main thread alive, or the process will exit.
            await new Promise((resolve) => setTimeout(resolve, 10000))
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}

export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
```

```
    }
}

// starting point
const subscribeToTopic = new SubscribeToTopic();
```

Esempi

Utilizza i seguenti esempi per imparare a utilizzare il servizio IPC di pubblicazione/sottoscrizione nei tuoi componenti.

Esempio di publisher publish/subscribe (Java, client IPC V1)

La seguente ricetta di esempio consente al componente di pubblicare su tutti gli argomenti.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubPublisherJava",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that publishes messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubPublisherJava:pubsub:1": {
            "policyDescription": "Allows access to publish to all topics.",
            "operations": [
              "aws.greengrass#PublishToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
```

```
      "Lifecycle": {
        "run": "java -jar {artifacts:path}/PubSubPublisher.jar"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubPublisherJava
ComponentVersion: '1.0.0'
ComponentDescription: A component that publishes messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        'com.example.PubSubPublisherJava:pubsub:1':
          policyDescription: Allows access to publish to all topics.
          operations:
            - 'aws.greengrass#PublishToTopic'
          resources:
            - '*'
Manifests:
  - Lifecycle:
      run: |-
        java -jar {artifacts:path}/PubSubPublisher.jar
```

L'applicazione Java di esempio seguente mostra come utilizzare il servizio IPC publish/subscribe per pubblicare messaggi su altri componenti.

```
/* Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: Apache-2.0 */

package com.example.ipc.pubsub;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.model.*;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;
```

```
import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class PubSubPublisher {

    public static void main(String[] args) {
        String message = "Hello from the pub/sub publisher (Java).";
        String topic = "test/topic/java";

        try (EventStreamRPCConnection eventStreamRPCConnection =
 IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient = new
 GreengrassCoreIPCClient(eventStreamRPCConnection);

            while (true) {
                PublishToTopicRequest publishRequest = new PublishToTopicRequest();
                PublishMessage publishMessage = new PublishMessage();
                BinaryMessage binaryMessage = new BinaryMessage();
                binaryMessage.setMessage(message.getBytes(StandardCharsets.UTF_8));
                publishMessage.setBinaryMessage(binaryMessage);
                publishRequest.setPublishMessage(publishMessage);
                publishRequest.setTopic(topic);
                CompletableFuture<PublishToTopicResponse> futureResponse = ipcClient
                        .publishToTopic(publishRequest,
 Optional.empty()).getResponse();

                try {
                    futureResponse.get(10, TimeUnit.SECONDS);
                    System.out.println("Successfully published to topic: " + topic);
                } catch (TimeoutException e) {
                    System.err.println("Timeout occurred while publishing to topic: " +
 topic);
                } catch (ExecutionException e) {
                    if (e.getCause() instanceof UnauthorizedError) {
                        System.err.println("Unauthorized error while publishing to
 topic: " + topic);
                    } else {
                        System.err.println("Execution exception while publishing to
 topic: " + topic);
                    }
```

```
                    throw e;
                }
                Thread.sleep(5000);
            }
        } catch (InterruptedException e) {
            System.out.println("Publisher interrupted.");
        } catch (Exception e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

Esempio di abbonato publish/subscribe (Java, client IPC V1)

La seguente ricetta di esempio consente al componente di sottoscrivere tutti gli argomenti.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubSubscriberJava",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that subscribes to messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubSubscriberJava:pubsub:1": {
            "policyDescription": "Allows access to subscribe to all topics.",
            "operations": [
              "aws.greengrass#SubscribeToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
```

```
    {
      "Lifecycle": {
        "run": "java -jar {artifacts:path}/PubSubSubscriber.jar"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubSubscriberJava
ComponentVersion: '1.0.0'
ComponentDescription: A component that subscribes to messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        'com.example.PubSubSubscriberJava:pubsub:1':
          policyDescription: Allows access to subscribe to all topics.
          operations:
            - 'aws.greengrass#SubscribeToTopic'
          resources:
            - '*'
Manifests:
  - Lifecycle:
      run: |-
        java -jar {artifacts:path}/PubSubSubscriber.jar
```

L'applicazione Java di esempio seguente mostra come utilizzare il servizio IPC publish/subscribe per sottoscrivere i messaggi di altri componenti.

```
/* Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
 * SPDX-License-Identifier: Apache-2.0 */

package com.example.ipc.pubsub;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.SubscribeToTopicResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.SubscribeToTopicRequest;
```

```
import software.amazon.awssdk.aws.greengrass.model.SubscribeToTopicResponse;
import software.amazon.awssdk.aws.greengrass.model.SubscriptionResponseMessage;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;
import software.amazon.awssdk.eventstreamrpc.StreamResponseHandler;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class PubSubSubscriber {

    public static void main(String[] args) {
        String topic = "test/topic/java";

        try (EventStreamRPCConnection eventStreamRPCConnection =
 IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient = new
 GreengrassCoreIPCClient(eventStreamRPCConnection);

            SubscribeToTopicRequest subscribeRequest = new SubscribeToTopicRequest();
            subscribeRequest.setTopic(topic);
            SubscribeToTopicResponseHandler operationResponseHandler = ipcClient
                    .subscribeToTopic(subscribeRequest, Optional.of(new
 SubscribeResponseHandler()));
            CompletableFuture<SubscribeToTopicResponse> futureResponse =
 operationResponseHandler.getResponse();

            try {
                futureResponse.get(10, TimeUnit.SECONDS);
                System.out.println("Successfully subscribed to topic: " + topic);
            } catch (TimeoutException e) {
                System.err.println("Timeout occurred while subscribing to topic: " +
 topic);
                throw e;
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.println("Unauthorized error while subscribing to topic:
 " + topic);
                } else {
```

```
                    System.err.println("Execution exception while subscribing to topic:
" + topic);
                }
                throw e;
            }

            // Keep the main thread alive, or the process will exit.
            try {
                while (true) {
                    Thread.sleep(10000);
                }
            } catch (InterruptedException e) {
                System.out.println("Subscribe interrupted.");
            }
        } catch (Exception e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    private static class SubscribeResponseHandler implements
StreamResponseHandler<SubscriptionResponseMessage> {

        @Override
        public void onStreamEvent(SubscriptionResponseMessage
subscriptionResponseMessage) {
            try {
                String message = new
String(subscriptionResponseMessage.getBinaryMessage()
                        .getMessage(), StandardCharsets.UTF_8);
                System.out.println("Received new message: " + message);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        @Override
        public boolean onStreamError(Throwable error) {
            System.err.println("Received a stream error.");
            error.printStackTrace();
            return false; // Return true to close stream, false to keep stream open.
        }
```

```
        @Override
        public void onStreamClosed() {
            System.out.println("Subscribe to topic stream closed.");
        }
    }
}
```

Esempio di editore publish/subscribe (Python, client IPC V1)

La seguente ricetta di esempio consente al componente di pubblicare su tutti gli argomenti.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubPublisherPython",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that publishes messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubPublisherPython:pubsub:1": {
            "policyDescription": "Allows access to publish to all topics.",
            "operations": [
              "aws.greengrass#PublishToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk",
        "run": "python3 -u {artifacts:path}/pubsub_publisher.py"
```

```
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk",
        "run": "py -3 -u {artifacts:path}/pubsub_publisher.py"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubPublisherPython
ComponentVersion: 1.0.0
ComponentDescription: A component that publishes messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        com.example.PubSubPublisherPython:pubsub:1:
          policyDescription: Allows access to publish to all topics.
          operations:
            - aws.greengrass#PublishToTopic
          resources:
            - "*"
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: python3 -m pip install --user awsiotsdk
      run: python3 -u {artifacts:path}/pubsub_publisher.py
  - Platform:
      os: windows
    Lifecycle:
      install: py -3 -m pip install --user awsiotsdk
      run: py -3 -u {artifacts:path}/pubsub_publisher.py
```

L'esempio seguente di applicazione Python dimostra come utilizzare il servizio IPC publish/subscribe per pubblicare messaggi su altri componenti.

```
import concurrent.futures
import sys
import time
import traceback

import awsiot.greengrasscoreipc
from awsiot.greengrasscoreipc.model import (
    PublishToTopicRequest,
    PublishMessage,
    BinaryMessage,
    UnauthorizedError
)


topic = "test/topic/python"
message = "Hello from the pub/sub publisher (Python)."
TIMEOUT = 10

try:
    ipc_client = awsiot.greengrasscoreipc.connect()

    while True:
        request = PublishToTopicRequest()
        request.topic = topic
        publish_message = PublishMessage()
        publish_message.binary_message = BinaryMessage()
        publish_message.binary_message.message = bytes(message, "utf-8")
        request.publish_message = publish_message
        operation = ipc_client.new_publish_to_topic()
        operation.activate(request)
        future_response = operation.get_response()

        try:
            future_response.result(TIMEOUT)
            print('Successfully published to topic: ' + topic)
        except concurrent.futures.TimeoutError:
            print('Timeout occurred while publishing to topic: ' + topic,
 file=sys.stderr)
        except UnauthorizedError as e:
```

```
            print('Unauthorized error while publishing to topic: ' + topic,
 file=sys.stderr)
            raise e
        except Exception as e:
            print('Exception while publishing to topic: ' + topic, file=sys.stderr)
            raise e
        time.sleep(5)
except InterruptedError:
    print('Publisher interrupted.')
except Exception:
    print('Exception occurred when using IPC.', file=sys.stderr)
    traceback.print_exc()
    exit(1)
```

Esempio di sottoscrittore publish/subscribe (Python, client IPC V1)

La seguente ricetta di esempio consente al componente di sottoscrivere tutti gli argomenti.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubSubscriberPython",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that subscribes to messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubSubscriberPython:pubsub:1": {
            "policyDescription": "Allows access to subscribe to all topics.",
            "operations": [
              "aws.greengrass#SubscribeToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
```

```
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk",
        "run": "python3 -u {artifacts:path}/pubsub_subscriber.py"
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk",
        "run": "py -3 -u {artifacts:path}/pubsub_subscriber.py"
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubSubscriberPython
ComponentVersion: 1.0.0
ComponentDescription: A component that subscribes to messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        com.example.PubSubSubscriberPython:pubsub:1:
          policyDescription: Allows access to subscribe to all topics.
          operations:
            - aws.greengrass#SubscribeToTopic
          resources:
            - "*"
Manifests:
  - Platform:
      os: linux
    Lifecycle:
```

```
      install: python3 -m pip install --user awsiotsdk
      run: python3 -u {artifacts:path}/pubsub_subscriber.py
  - Platform:
      os: windows
    Lifecycle:
      install: py -3 -m pip install --user awsiotsdk
      run: py -3 -u {artifacts:path}/pubsub_subscriber.py
```

L'esempio seguente di applicazione Python dimostra come utilizzare il servizio IPC publish/subscribe per sottoscrivere messaggi ad altri componenti.

```
import concurrent.futures
import sys
import time
import traceback

import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import (
    SubscribeToTopicRequest,
    SubscriptionResponseMessage,
    UnauthorizedError
)

topic = "test/topic/python"
TIMEOUT = 10


class StreamHandler(client.SubscribeToTopicStreamHandler):
    def __init__(self):
        super().__init__()

    def on_stream_event(self, event: SubscriptionResponseMessage) -> None:
        try:
            message = str(event.binary_message.message, "utf-8")
            print("Received new message: " + message)
        except:
            traceback.print_exc()

    def on_stream_error(self, error: Exception) -> bool:
        print("Received a stream error.", file=sys.stderr)
        traceback.print_exc()
```

```
        return False  # Return True to close stream, False to keep stream open.

    def on_stream_closed(self) -> None:
        print('Subscribe to topic stream closed.')


try:
    ipc_client = awsiot.greengrasscoreipc.connect()

    request = SubscribeToTopicRequest()
    request.topic = topic
    handler = StreamHandler()
    operation = ipc_client.new_subscribe_to_topic(handler)
    operation.activate(request)
    future_response = operation.get_response()

    try:
        future_response.result(TIMEOUT)
        print('Successfully subscribed to topic: ' + topic)
    except concurrent.futures.TimeoutError as e:
        print('Timeout occurred while subscribing to topic: ' + topic,
 file=sys.stderr)
        raise e
    except UnauthorizedError as e:
        print('Unauthorized error while subscribing to topic: ' + topic,
 file=sys.stderr)
        raise e
    except Exception as e:
        print('Exception while subscribing to topic: ' + topic, file=sys.stderr)
        raise e

    # Keep the main thread alive, or the process will exit.
    try:
        while True:
            time.sleep(10)
    except InterruptedError:
        print('Subscribe interrupted.')
except Exception:
    print('Exception occurred when using IPC.', file=sys.stderr)
    traceback.print_exc()
    exit(1)
```

Esempio publish/subscribe publisher (C++)

La seguente ricetta di esempio consente al componente di pubblicare su tutti gli argomenti.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubPublisherCpp",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that publishes messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubPublisherCpp:pubsub:1": {
            "policyDescription": "Allows access to publish to all topics.",
            "operations": [
              "aws.greengrass#PublishToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "{artifacts:path}/greengrassv2_pubsub_publisher"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.PubSubPublisherCpp/1.0.0/greengrassv2_pubsub_publisher",
          "Permission": {
            "Execute": "OWNER"
          }
        }
      ]
    }
```

```
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubPublisherCpp
ComponentVersion: 1.0.0
ComponentDescription: A component that publishes messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        com.example.PubSubPublisherCpp:pubsub:1:
          policyDescription: Allows access to publish to all topics.
          operations:
            - aws.greengrass#PublishToTopic
          resources:
            - "*"
Manifests:
  - Lifecycle:
      run: "{artifacts:path}/greengrassv2_pubsub_publisher"
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/com.example.PubSubPublisherCpp/1.0.0/
greengrassv2_pubsub_publisher
        Permission:
          Execute: OWNER
```

L'applicazione C++ di esempio seguente mostra come utilizzare il servizio IPC publish/subscribe per pubblicare messaggi su altri componenti.

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
```

```
    void OnConnectCallback() override {
        std::cout << "OnConnectCallback" << std::endl;
    }

    void OnDisconnectCallback(RpcError error) override {
        std::cout << "OnDisconnectCallback: " << error.StatusToString() << std::endl;
        exit(-1);
    }

    bool OnErrorCallback(RpcError error) override {
        std::cout << "OnErrorCallback: " << error.StatusToString() << std::endl;
        return true;
    }
};

int main() {
    String message("Hello from the pub/sub publisher (C++).");
    String topic("test/topic/cpp");
    int timeout = 10;

    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    while (true) {
        PublishToTopicRequest request;
        Vector<uint8_t> messageData({message.begin(), message.end()});
        BinaryMessage binaryMessage;
        binaryMessage.SetMessage(messageData);
        PublishMessage publishMessage;
        publishMessage.SetBinaryMessage(binaryMessage);
        request.SetTopic(topic);
        request.SetPublishMessage(publishMessage);

        auto operation = ipcClient.NewPublishToTopic();
```

```
        auto activate = operation->Activate(request, nullptr);
        activate.wait();

        auto responseFuture = operation->GetResult();
        if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
            std::cerr << "Operation timed out while waiting for response from
 Greengrass Core." << std::endl;
            exit(-1);
        }

        auto response = responseFuture.get();
        if (response) {
            std::cout << "Successfully published to topic: " << topic << std::endl;
        } else {
            // An error occurred.
            std::cout << "Failed to publish to topic: " << topic << std::endl;
            auto errorType = response.GetResultType();
            if (errorType == OPERATION_ERROR) {
                auto *error = response.GetOperationError();
                std::cout << "Operation error: " << error->GetMessage().value() <<
 std::endl;
            } else {
                std::cout << "RPC error: " << response.GetRpcError() << std::endl;
            }
            exit(-1);
        }

        std::this_thread::sleep_for(std::chrono::seconds(5));
    }

    return 0;
}
```

Esempio di abbonato publish/subscribe (C++)

La seguente ricetta di esempio consente al componente di sottoscrivere tutti gli argomenti.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PubSubSubscriberCpp",
  "ComponentVersion": "1.0.0",
```

```
  "ComponentDescription": "A component that subscribes to messages.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.pubsub": {
          "com.example.PubSubSubscriberCpp:pubsub:1": {
            "policyDescription": "Allows access to subscribe to all topics.",
            "operations": [
              "aws.greengrass#SubscribeToTopic"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "{artifacts:path}/greengrassv2_pub_sub_subscriber"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.PubSubSubscriberCpp/1.0.0/greengrassv2_pub_sub_subscriber",
          "Permission": {
            "Execute": "OWNER"
          }
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PubSubSubscriberCpp
ComponentVersion: 1.0.0
```

```
ComponentDescription: A component that subscribes to messages.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.pubsub:
        com.example.PubSubSubscriberCpp:pubsub:1:
          policyDescription: Allows access to subscribe to all topics.
          operations:
            - aws.greengrass#SubscribeToTopic
          resources:
            - "*"
Manifests:
  - Lifecycle:
      run: "{artifacts:path}/greengrassv2_pub_sub_subscriber"
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.PubSubSubscriberCpp/1.0.0/greengrassv2_pub_sub_subscriber
        Permission:
          Execute: OWNER
```

L'applicazione C++ di esempio seguente mostra come utilizzare il servizio IPC publish/subscribe per sottoscrivere i messaggi di altri componenti.

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class SubscribeResponseHandler : public SubscribeToTopicStreamHandler {
    public:
        virtual ~SubscribeResponseHandler() {}

    private:
        void OnStreamEvent(SubscriptionResponseMessage *response) override {
            auto jsonMessage = response->GetJsonMessage();
            if (jsonMessage.has_value() &&
 jsonMessage.value().GetMessage().has_value()) {
```

```
                auto messageString =
 jsonMessage.value().GetMessage().value().View().WriteReadable();
                std::cout << "Received new message: " << messageString << std::endl;
            } else {
                auto binaryMessage = response->GetBinaryMessage();
                if (binaryMessage.has_value() &&
 binaryMessage.value().GetMessage().has_value()) {
                    auto messageBytes = binaryMessage.value().GetMessage().value();
                    std::string messageString(messageBytes.begin(),
 messageBytes.end());
                    std::cout << "Received new message: " << messageString <<
 std::endl;
                }
            }
        }

        bool OnStreamError(OperationError *error) override {
            std::cout << "Received an operation error: ";
            if (error->GetMessage().has_value()) {
                std::cout << error->GetMessage().value();
            }
            std::cout << std::endl;
            return false; // Return true to close stream, false to keep stream open.
        }

        void OnStreamClosed() override {
            std::cout << "Subscribe to topic stream closed." << std::endl;
        }
};

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        std::cout << "OnConnectCallback" << std::endl;
    }

    void OnDisconnectCallback(RpcError error) override {
        std::cout << "OnDisconnectCallback: " << error.StatusToString() << std::endl;
        exit(-1);
    }

    bool OnErrorCallback(RpcError error) override {
        std::cout << "OnErrorCallback: " << error.StatusToString() << std::endl;
        return true;
    }
```

```
};

int main() {
    String topic("test/topic/cpp");
    int timeout = 10;

    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    SubscribeToTopicRequest request;
    request.SetTopic(topic);
    auto streamHandler = MakeShared<SubscribeResponseHandler>(DefaultAllocator());
    auto operation = ipcClient.NewSubscribeToTopic(streamHandler);
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
 Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (response) {
        std::cout << "Successfully subscribed to topic: " << topic << std::endl;
    } else {
        // An error occurred.
        std::cout << "Failed to subscribe to topic: " << topic << std::endl;
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
```

```
            std::cout << "Operation error: " << error->GetMessage().value() <<
 std::endl;
        } else {
            std::cout << "RPC error: " << response.GetRpcError() << std::endl;
        }
        exit(-1);
    }

    // Keep the main thread alive, or the process will exit.
    while (true) {
        std::this_thread::sleep_for(std::chrono::seconds(10));
    }

    operation->Close();
    return 0;
}
```

AWS IoT Core Pubblicare/sottoscrivere messaggi MQTT

Il servizio IPC di messaggistica AWS IoT Core MQTT consente di inviare e ricevere messaggi MQTT da e verso. AWS IoT Core I componenti possono pubblicare messaggi AWS IoT Core e sottoscrivere argomenti per agire sui messaggi MQTT provenienti da altre fonti. Per ulteriori informazioni sull' AWS IoT Core implementazione di MQTT, consultate MQTT nella Developer Guide.AWS IoT Core

> **Note**
>
> Questo servizio IPC di messaggistica MQTT consente di scambiare messaggi con. AWS IoT Core Per ulteriori informazioni su come scambiare messaggi tra componenti, vedere. Pubblicare/sottoscrivere messaggi locali

Argomenti

- Versioni SDK minime
- Autorizzazione
- PublishToIoTCore
- SubscribeToIoTCore
- Esempi

Versioni SDK minime

La tabella seguente elenca le versioni minime da utilizzare per pubblicare e sottoscrivere messaggi MQTT da e verso. SDK per dispositivi AWS IoT AWS IoT Core

SDK	Versione minima	
SDK per dispositivi AWS IoT per Java v2	v1.2.10	
SDK per dispositivi AWS IoT per Python v2	v1.5.3	
SDK per dispositivi AWS IoT per C++ v2	v1.17.0	
SDK per dispositivi AWS IoT per v2 JavaScript	v1.12.0	

Autorizzazione

Per utilizzare la messaggistica AWS IoT Core MQTT in un componente personalizzato, è necessario definire politiche di autorizzazione che consentano al componente di inviare e ricevere messaggi su argomenti. Per informazioni sulla definizione delle politiche di autorizzazione, vedereAutorizza i componenti a eseguire operazioni IPC.

Le politiche di autorizzazione per la messaggistica AWS IoT Core MQTT hanno le seguenti proprietà.

Identificatore del servizio IPC: `aws.greengrass.ipc.mqttproxy`

Operazione	Descrizione	Risorse
`aws.greengrass#PublishToIoTCore`	Consente a un componente di pubblicare messaggi AWS IoT Core sugli argomenti MQTT specificati.	Una stringa di argomenti, ad esempio`test/topic` , o `*` per consentire l'accesso a tutti gli argomenti. È possibile utilizzare i caratteri jolly degli

Operazione	Descrizione	Risorse
		argomenti MQTT (#and+) per abbinare più risorse.
`aws.greengrass#SubscribeToIoTCore`	Consente a un componente di sottoscrivere i messaggi provenienti dagli AWS IoT Core argomenti specificati.	Una stringa di argomento, ad esempio`test/topic` , o `*` per consentire l'accesso a tutti gli argomenti. È possibile utilizzare i caratteri jolly degli argomenti MQTT (#and+) per abbinare più risorse.
*	Consente a un componente di pubblicare e sottoscrivere messaggi AWS IoT Core MQTT per gli argomenti specificati.	Una stringa di argomenti, ad esempio`test/topic` , o `*` per consentire l'accesso a tutti gli argomenti. È possibile utilizzare i caratteri jolly degli argomenti MQTT (#and+) per abbinare più risorse.

I caratteri jolly MQTT nelle politiche di autorizzazione MQTT AWS IoT Core

È possibile utilizzare i caratteri jolly MQTT nelle AWS IoT Core politiche di autorizzazione MQTT IPC. I componenti possono pubblicare e sottoscrivere argomenti che corrispondono al filtro degli argomenti consentito in una politica di autorizzazione. Ad esempio, se la politica di autorizzazione di un componente concede l'accesso a`test/topic/#`, il componente può sottoscrivere`test/topic/#`, pubblicare e sottoscrivere`test/topic/filter`.

Variabili Recipe nelle politiche di autorizzazione AWS IoT Core MQTT

Se si utilizza la versione 2.6.0 o successiva del nucleo Greengrass, è possibile utilizzare la variabile recipe nelle politiche di autorizzazione. `{iot:thingName}` Questa funzionalità consente di configurare un'unica politica di autorizzazione per un gruppo di dispositivi principali, in cui ogni dispositivo principale può accedere solo agli argomenti che contengono il proprio nome. Ad esempio, è possibile consentire a un componente l'accesso alla seguente risorsa tematica.

```
devices/{iot:thingName}/messages
```

Per ulteriori informazioni, consulta Variabili di ricetta e Usa le variabili di ricetta negli aggiornamenti di fusione.

Esempi di politiche di autorizzazione

Puoi fare riferimento ai seguenti esempi di politiche di autorizzazione per aiutarti a configurare le politiche di autorizzazione per i tuoi componenti.

Example Esempio di politica di autorizzazione con accesso illimitato

Il seguente esempio di politica di autorizzazione consente a un componente di pubblicare e sottoscrivere tutti gli argomenti.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "com.example.MyIoTCorePubSubComponent:mqttproxy:1": {
        "policyDescription": "Allows access to publish/subscribe to all topics.",
        "operations": [
          "aws.greengrass#PublishToIoTCore",
          "aws.greengrass#SubscribeToIoTCore"
        ],
        "resources": [
          "*"
        ]
      }
    }
  }
}
```

YAML

```
---
accessControl:
  aws.greengrass.ipc.mqttproxy:
    com.example.MyIoTCorePubSubComponent:mqttproxy:1:
      policyDescription: Allows access to publish/subscribe to all topics.
      operations:
```

```
        - aws.greengrass#PublishToIoTCore
        - aws.greengrass#SubscribeToIoTCore
      resources:
        - "*"
```

Example Esempio di politica di autorizzazione con accesso limitato

Il seguente esempio di politica di autorizzazione consente a un componente di pubblicare e sottoscrivere due argomenti denominati `factory/1/events` e`factory/1/actions`.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "com.example.MyIoTCorePubSubComponent:mqttproxy:1": {
        "policyDescription": "Allows access to publish/subscribe to factory 1
 topics.",
        "operations": [
          "aws.greengrass#PublishToIoTCore",
          "aws.greengrass#SubscribeToIoTCore"
        ],
        "resources": [
          "factory/1/actions",
          "factory/1/events"
        ]
      }
    }
  }
}
```

YAML

```
---
accessControl:
  aws.greengrass.ipc.mqttproxy:
    "com.example.MyIoTCorePubSubComponent:mqttproxy:1":
      policyDescription: Allows access to publish/subscribe to factory 1 topics.
      operations:
        - aws.greengrass#PublishToIoTCore
        - aws.greengrass#SubscribeToIoTCore
      resources:
```

```
        - factory/1/actions
        - factory/1/events
```

Example Esempio di politica di autorizzazione per un gruppo di dispositivi principali

> **Important**
>
> Questo esempio utilizza una funzionalità disponibile per la versione 2.6.0 e successive del componente Greengrass nucleus. Greengrass nucleus v2.6.0 aggiunge il supporto per la maggior parte delle variabili di ricetta, ad esempio nelle configurazioni dei componenti. `{iot:thingName}`

Il seguente esempio di politica di autorizzazione consente a un componente di pubblicare e sottoscrivere un argomento che contiene il nome del dispositivo principale che esegue il componente.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "com.example.MyIoTCorePubSubComponent:mqttproxy:1": {
        "policyDescription": "Allows access to publish/subscribe to all topics.",
        "operations": [
          "aws.greengrass#PublishToIoTCore",
          "aws.greengrass#SubscribeToIoTCore"
        ],
        "resources": [
          "factory/1/devices/{iot:thingName}/controls"
        ]
      }
    }
  }
}
```

YAML

```
---
accessControl:
  aws.greengrass.ipc.mqttproxy:
    "com.example.MyIoTCorePubSubComponent:mqttproxy:1":
```

```
      policyDescription: Allows access to publish/subscribe to all topics.
      operations:
        - aws.greengrass#PublishToIoTCore
        - aws.greengrass#SubscribeToIoTCore
      resources:
        - factory/1/devices/{iot:thingName}/controls
```

PublishToIoTCore

Pubblica un messaggio MQTT AWS IoT Core su un argomento.

Quando si pubblicano messaggi MQTT su AWS IoT Core, esiste una quota di 100 transazioni al secondo. Se si supera questa quota, i messaggi vengono messi in coda per l'elaborazione sul dispositivo Greengrass. È inoltre prevista una quota di 512 Kb di dati al secondo e una quota a livello di account di 20.000 pubblicazioni al secondo (2.000 in alcune). Regioni AWS Per ulteriori informazioni sui limiti e le quote del broker di messaggi MQTT in AWS IoT Core, vedere AWS IoT Core Message Broker and Protocol Limits and Protocol.

Se superi queste quote, il dispositivo Greengrass limita la pubblicazione dei messaggi a. AWS IoT Core I messaggi vengono archiviati in uno spooler in memoria. Per impostazione predefinita, la memoria allocata allo spooler è 2,5 Mb. Se lo spooler si riempie, i nuovi messaggi vengono rifiutati. È possibile aumentare le dimensioni dello spooler. Per ulteriori informazioni, consulta Configurazione nella documentazione Nucleo Greengrass. Per evitare di riempire lo spooler e di dover aumentare la memoria allocata, limita le richieste di pubblicazione a non più di 100 richieste al secondo.

Quando la tua applicazione deve inviare messaggi a una velocità maggiore o messaggi più grandi, prendi in considerazione l'utilizzo di Stream manager per inviare messaggi a Kinesis Data Streams. Il componente stream manager è progettato per trasferire grandi volumi di dati a. Cloud AWS Per ulteriori informazioni, consulta Gestisci i flussi di dati sui dispositivi core Greengrass.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`topicName`(Python:) `topic_name`

L'argomento su cui pubblicare il messaggio.

`qos`

Il QoS MQTT da usare. Questo enum`QOS`, ha i seguenti valori:

- `AT_MOST_ONCE`— QoS 0. Il messaggio MQTT viene recapitato al massimo una volta.
- `AT_LEAST_ONCE`— QoS 1. Il messaggio MQTT viene recapitato almeno una volta.

`payload`

(Facoltativo) Il payload del messaggio sotto forma di blob.

Le seguenti funzionalità sono disponibili per la versione 2.10.0 e successive se si utilizza MQTT 5. Nucleo Greengrass Queste funzionalità vengono ignorate quando si utilizza MQTT 3.1.1. La tabella seguente elenca la versione minima dell'SDK del AWS IoT dispositivo da utilizzare per accedere a queste funzionalità.

SDK	Versione minima
SDK per dispositivi AWS IoT per Python v2	v1.15.0
SDK per dispositivi AWS IoT per Java v2	v1.13.0
SDK per dispositivi AWS IoT per C++ v2	v1.24.0
SDK per dispositivi AWS IoT per JavaScript v2	v1.13.0

`payloadFormat`

(Facoltativo) Il formato del payload del messaggio. Se non si imposta il`payloadFormat`, si presume che il tipo sia. `BYTES` L'enum ha i seguenti valori:

- `BYTES`— Il contenuto del payload è un blob binario.
- `UTF8`— Il contenuto del payload è una stringa di caratteri UTF8.

`retain`

(Facoltativo) Indica se impostare l'opzione di conservazione MQTT su durante la pubblicazione. `true`

`userProperties`

(Facoltativo) Un elenco di oggetti specifici dell'applicazione da inviare`UserProperty`. L'`UserProperty`oggetto è definito come segue:

```
UserProperty:
```

```
  key: string
  value: string
```

`messageExpiryIntervalSeconds`

(Facoltativo) Il numero di secondi prima che il messaggio scada e venga eliminato dal server. Se questo valore non è impostato, il messaggio non scade.

`correlationData`

(Facoltativo) Informazioni aggiunte alla richiesta che possono essere utilizzate per associare una richiesta a una risposta.

`responseTopic`

(Facoltativo) L'argomento da utilizzare per il messaggio di risposta.

`contentType`

(Facoltativo) Un identificatore specifico dell'applicazione del tipo di contenuto del messaggio.

Risposta

Questa operazione non fornisce alcuna informazione nella sua risposta.

Esempi

Gli esempi seguenti mostrano come chiamare questa operazione nel codice componente personalizzato.

Java (IPC client V2)

Example Esempio: pubblicare un messaggio

```
package com.aws.greengrass.docs.samples.ipc;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClientV2;
import software.amazon.awssdk.aws.greengrass.model.PublishToIoTCoreRequest;
import software.amazon.awssdk.aws.greengrass.model.QOS;
import java.nio.charset.StandardCharsets;

public class PublishToIoTCore {
```

```
    public static void main(String[] args) {
        String topic = args[0];
        String message = args[1];
        QOS qos = QOS.get(args[2]);

        try (GreengrassCoreIPCClientV2 ipcClientV2 =
 GreengrassCoreIPCClientV2.builder().build()) {
            ipcClientV2.publishToIoTCore(new PublishToIoTCoreRequest()
                    .withTopicName(topic)
                    .withPayload(message.getBytes(StandardCharsets.UTF_8))
                    .withQos(qos));
            System.out.println("Successfully published to topic: " + topic);
        } catch (Exception e) {
            System.err.println("Exception occurred.");
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

Python (IPC client V2)

Example Esempio: pubblicare un messaggio

> **Note**
>
> Questo esempio presuppone che si stia utilizzando la versione 1.5.4 o successiva di for SDK per dispositivi AWS IoT Python v2.

```
import awsiot.greengrasscoreipc.clientv2 as clientV2

topic = 'my/topic'
qos = '1'
payload = 'Hello, World'

ipc_client = clientV2.GreengrassCoreIPCClientV2()
resp = ipc_client.publish_to_iot_core(topic_name=topic, qos=qos, payload=payload)
ipc_client.close()
```

Java (IPC client V1)

Example Esempio: pubblicare un messaggio

> **Note**
>
> Questo esempio utilizza una `IPCUtils` classe per creare una connessione al servizio AWS IoT Greengrass Core IPC. Per ulteriori informazioni, consulta Connect al servizio AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.PublishToIoTCoreResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.PublishToIoTCoreRequest;
import software.amazon.awssdk.aws.greengrass.model.PublishToIoTCoreResponse;
import software.amazon.awssdk.aws.greengrass.model.QOS;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class PublishToIoTCore {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        String topic = args[0];
        String message = args[1];
        QOS qos = QOS.get(args[2]);
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            PublishToIoTCoreResponseHandler responseHandler =
```

```
                    PublishToIoTCore.publishBinaryMessageToTopic(ipcClient, topic,
 message, qos);
            CompletableFuture<PublishToIoTCoreResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                System.out.println("Successfully published to topic: " + topic);
            } catch (TimeoutException e) {
                System.err.println("Timeout occurred while publishing to topic: " +
 topic);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.println("Unauthorized error while publishing to
 topic: " + topic);
                } else {
                    throw e;
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static PublishToIoTCoreResponseHandler
 publishBinaryMessageToTopic(GreengrassCoreIPCClient greengrassCoreIPCClient, String
 topic, String message, QOS qos) {
        PublishToIoTCoreRequest publishToIoTCoreRequest = new
 PublishToIoTCoreRequest();
        publishToIoTCoreRequest.setTopicName(topic);

 publishToIoTCoreRequest.setPayload(message.getBytes(StandardCharsets.UTF_8));
        publishToIoTCoreRequest.setQos(qos);
        return greengrassCoreIPCClient.publishToIoTCore(publishToIoTCoreRequest,
 Optional.empty());
    }
}
```

Python (IPC client V1)

Example Esempio: pubblicare un messaggio

> **Note**
>
> Questo esempio presuppone che si stia utilizzando la versione 1.5.4 o successiva di for SDK per dispositivi AWS IoT Python v2.

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import (
    QOS,
    PublishToIoTCoreRequest
)

TIMEOUT = 10

ipc_client = awsiot.greengrasscoreipc.connect()

topic = "my/topic"
message = "Hello, World"
qos = QOS.AT_LEAST_ONCE

request = PublishToIoTCoreRequest()
request.topic_name = topic
request.payload = bytes(message, "utf-8")
request.qos = qos
operation = ipc_client.new_publish_to_iot_core()
operation.activate(request)
future_response = operation.get_response()
future_response.result(TIMEOUT)
```

C++

Example Esempio: pubblicare un messaggio

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>
```

```
using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        // Handle connection to IPC service.
    }

    void OnDisconnectCallback(RpcError error) override {
        // Handle disconnection from IPC service.
    }

    bool OnErrorCallback(RpcError error) override {
        // Handle IPC service connection error.
        return true;
    }
};

int main() {
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    String message("Hello, World!");
    String topic("my/topic");
    QOS qos = QOS_AT_MOST_ONCE;
    int timeout = 10;

    PublishToIoTCoreRequest request;
    Vector<uint8_t> messageData({message.begin(), message.end()});
    request.SetTopicName(topic);
    request.SetPayload(messageData);
    request.SetQos(qos);

    auto operation = ipcClient.NewPublishToIoTCore();
```

```
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
 Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (!response) {
        // Handle error.
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            (void)error;
            // Handle operation error.
        } else {
            // Handle RPC error.
        }
    }

    return 0;
}
```

JavaScript

Example Esempio: pubblicare un messaggio

```
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";
import {QOS, PublishToIoTCoreRequest} from "aws-iot-device-sdk-v2/dist/
greengrasscoreipc/model";

class PublishToIoTCore {
    private ipcClient: greengrasscoreipc.Client
    private readonly topic: string;

    constructor() {
        // define your own constructor, e.g.
        this.topic = "<define_your_topic>";
        this.publishToIoTCore().then(r => console.log("Started workflow"));
```

```
    }

    private async publishToIoTCore() {
        try {
            const request: PublishToIoTCoreRequest = {
                topicName: this.topic,
                qos: QOS.AT_LEAST_ONCE, // you can change this depending on your use
 case
            }

            this.ipcClient = await getIpcClient();

            await this.ipcClient.publishToIoTCore(request);
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}


export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

// starting point
const publishToIoTCore = new PublishToIoTCore();
```

SubscribeToIoTCore

Iscriviti ai messaggi MQTT da un AWS IoT Core argomento o da un filtro per argomento. Il software AWS IoT Greengrass Core rimuove gli abbonamenti quando il componente raggiunge la fine del suo ciclo di vita.

Questa operazione è un'operazione di sottoscrizione in cui ci si iscrive a un flusso di messaggi di eventi. Per utilizzare questa operazione, definite un gestore di risposte di flusso con funzioni che gestiscono i messaggi di evento, gli errori e la chiusura dei flussi. Per ulteriori informazioni, consulta Iscriviti ai flussi di eventi IPC.

Tipo di messaggio di evento: `IoTCoreMessage`

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`topicName`(Python:) `topic_name`

L'argomento a cui iscriversi. È possibile utilizzare i caratteri jolly degli argomenti MQTT (#and+) per sottoscrivere più argomenti.

`qos`

Il QoS MQTT da usare. Questo enum`QOS`, ha i seguenti valori:

- `AT_MOST_ONCE`— QoS 0. Il messaggio MQTT viene recapitato al massimo una volta.
- `AT_LEAST_ONCE`— QoS 1. Il messaggio MQTT viene recapitato almeno una volta.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`messages`

Il flusso di messaggi MQTT. Questo oggetto contiene `IoTCoreMessage` le seguenti informazioni:

`message`

Il messaggio MQTT. Questo oggetto contiene `MQTTMessage` le seguenti informazioni:

`topicName`(Python:) `topic_name`

L'argomento su cui è stato pubblicato il messaggio.

`payload`

(Facoltativo) Il payload del messaggio sotto forma di blob.

Le seguenti funzionalità sono disponibili per la versione 2.10.0 e successive se si utilizza MQTT 5. Nucleo Greengrass Queste funzionalità vengono ignorate quando si utilizza MQTT 3.1.1. La tabella seguente elenca la versione minima dell'SDK del AWS IoT dispositivo da utilizzare per accedere a queste funzionalità.

SDK	Versione minima
SDK per dispositivi AWS IoT per Python v2	v1.15.0
SDK per dispositivi AWS IoT per Java v2	v1.13.0
SDK per dispositivi AWS IoT per C++ v2	v1.24.0
SDK per dispositivi AWS IoT per JavaScript v2	v1.13.0

`payloadFormat`

(Facoltativo) Il formato del payload del messaggio. Se non si imposta il`payloadFormat`, si presume che il tipo sia. `BYTES` L'enum ha i seguenti valori:

- `BYTES`— Il contenuto del payload è un blob binario.
- `UTF8`— Il contenuto del payload è una stringa di caratteri UTF8.

`retain`

(Facoltativo) Indica se impostare l'opzione di conservazione MQTT su durante la pubblicazione. `true`

`userProperties`

(Facoltativo) Un elenco di oggetti specifici dell'applicazione da inviare`UserProperty`. L'`UserProperty`oggetto è definito come segue:

```
UserProperty:
  key: string
  value: string
```

`messageExpiryIntervalSeconds`

(Facoltativo) Il numero di secondi prima che il messaggio scada e venga eliminato dal server. Se questo valore non è impostato, il messaggio non scade.

`correlationData`

(Facoltativo) Informazioni aggiunte alla richiesta che possono essere utilizzate per associare una richiesta a una risposta.

`responseTopic`

(Facoltativo) L'argomento da utilizzare per il messaggio di risposta.

`contentType`

(Facoltativo) Un identificatore specifico dell'applicazione del tipo di contenuto del messaggio.

Esempi

Gli esempi seguenti mostrano come chiamare questa operazione nel codice componente personalizzato.

Java (IPC client V2)

Example Esempio: iscriversi ai messaggi

```
package com.aws.greengrass.docs.samples.ipc;

import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClientV2;
import software.amazon.awssdk.aws.greengrass.SubscribeToIoTCoreResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.QOS;
import software.amazon.awssdk.aws.greengrass.model.IoTCoreMessage;
import software.amazon.awssdk.aws.greengrass.model.SubscribeToIoTCoreRequest;
import software.amazon.awssdk.aws.greengrass.model.SubscribeToIoTCoreResponse;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.function.Consumer;
import java.util.function.Function;


public class SubscribeToIoTCore {
```

```
   public static void main(String[] args) {
       String topic = args[0];
       QOS qos = QOS.get(args[1]);

       Consumer<IoTCoreMessage> onStreamEvent = ioTCoreMessage ->
               System.out.printf("Received new message on topic %s: %s%n",
                       ioTCoreMessage.getMessage().getTopicName(),
                       new String(ioTCoreMessage.getMessage().getPayload(),
StandardCharsets.UTF_8));

       Optional<Function<Throwable, Boolean>> onStreamError =
               Optional.of(e -> {
                   System.err.println("Received a stream error.");
                   e.printStackTrace();
                   return false;
               });

       Optional<Runnable> onStreamClosed = Optional.of(() ->
               System.out.println("Subscribe to IoT Core stream closed."));

       try (GreengrassCoreIPCClientV2 ipcClientV2 =
GreengrassCoreIPCClientV2.builder().build()) {
           SubscribeToIoTCoreRequest request = new SubscribeToIoTCoreRequest()
                   .withTopicName(topic)
                   .withQos(qos);

           GreengrassCoreIPCClientV2.StreamingResponse<SubscribeToIoTCoreResponse,
SubscribeToIoTCoreResponseHandler>
                   streamingResponse = ipcClientV2.subscribeToIoTCore(request,
onStreamEvent, onStreamError, onStreamClosed);

           streamingResponse.getResponse();
           System.out.println("Successfully subscribed to topic: " + topic);

           // Keep the main thread alive, or the process will exit.
           while (true) {
               Thread.sleep(10000);
           }

           // To stop subscribing, close the stream.
           streamingResponse.getHandler().closeStream();
       } catch (InterruptedException e) {
           System.out.println("Subscribe interrupted.");
```

```
        } catch (Exception e) {
            System.err.println("Exception occurred.");
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

Python (IPC client V2)

Example Esempio: iscriversi ai messaggi

> **Note**
>
> Questo esempio presuppone che si stia utilizzando la versione 1.5.4 o successiva di for SDK per dispositivi AWS IoT Python v2.

```
import threading
import traceback

import awsiot.greengrasscoreipc.clientv2 as clientV2

topic = 'my/topic'
qos = '1'

def on_stream_event(event):
    try:
        topic_name = event.message.topic_name
        message = str(event.message.payload, 'utf-8')
        print(f'Received new message on topic {topic_name}:  {message}')
    except:
        traceback.print_exc()

def on_stream_error(error):
    # Return True to close stream, False to keep stream open.
    return True

def on_stream_closed():
    pass

ipc_client = clientV2.GreengrassCoreIPCClientV2()
```

```
resp, operation = ipc_client.subscribe_to_iot_core(
    topic_name=topic,
    qos=qos,
    on_stream_event=on_stream_event,
    on_stream_error=on_stream_error,
    on_stream_closed=on_stream_closed
)

# Keep the main thread alive, or the process will exit.
event = threading.Event()
event.wait()

# To stop subscribing, close the operation stream.
operation.close()
ipc_client.close()
```

Java (IPC client V1)

Example Esempio: iscriversi ai messaggi

> **Note**
>
> Questo esempio utilizza una `IPCUtils` classe per creare una connessione al servizio AWS IoT Greengrass Core IPC. Per ulteriori informazioni, consulta Connect al servizio AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.SubscribeToIoTCoreResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.*;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;
import software.amazon.awssdk.eventstreamrpc.StreamResponseHandler;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;
```

```
public class SubscribeToIoTCore {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        String topic = args[0];
        QOS qos = QOS.get(args[1]);
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            StreamResponseHandler<IoTCoreMessage> streamResponseHandler =
                    new SubscriptionResponseHandler();
            SubscribeToIoTCoreResponseHandler responseHandler =
                    SubscribeToIoTCore.subscribeToIoTCore(ipcClient, topic, qos,
                            streamResponseHandler);
            CompletableFuture<SubscribeToIoTCoreResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                System.out.println("Successfully subscribed to topic: " + topic);
            } catch (TimeoutException e) {
                System.err.println("Timeout occurred while subscribing to topic: " +
 topic);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.println("Unauthorized error while subscribing to
 topic: " + topic);
                } else {
                    throw e;
                }
            }

            // Keep the main thread alive, or the process will exit.
            try {
                while (true) {
                    Thread.sleep(10000);
                }
            } catch (InterruptedException e) {
                System.out.println("Subscribe interrupted.");
            }

            // To stop subscribing, close the stream.
```

```
            responseHandler.closeStream();
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static SubscribeToIoTCoreResponseHandler
subscribeToIoTCore(GreengrassCoreIPCClient greengrassCoreIPCClient, String topic,
QOS qos, StreamResponseHandler<IoTCoreMessage> streamResponseHandler) {
        SubscribeToIoTCoreRequest subscribeToIoTCoreRequest = new
SubscribeToIoTCoreRequest();
        subscribeToIoTCoreRequest.setTopicName(topic);
        subscribeToIoTCoreRequest.setQos(qos);
        return
greengrassCoreIPCClient.subscribeToIoTCore(subscribeToIoTCoreRequest,
                Optional.of(streamResponseHandler));
    }

    public static class SubscriptionResponseHandler implements
StreamResponseHandler<IoTCoreMessage> {

        @Override
        public void onStreamEvent(IoTCoreMessage ioTCoreMessage) {
            try {
                String topic = ioTCoreMessage.getMessage().getTopicName();
                String message = new
String(ioTCoreMessage.getMessage().getPayload(),
                        StandardCharsets.UTF_8);
                System.out.printf("Received new message on topic %s: %s%n", topic,
message);
            } catch (Exception e) {
                System.err.println("Exception occurred while processing subscription
response " +
                        "message.");
                e.printStackTrace();
            }
        }

        @Override
        public boolean onStreamError(Throwable error) {
```

```
            System.err.println("Received a stream error.");
            error.printStackTrace();
            return false;
        }

        @Override
        public void onStreamClosed() {
            System.out.println("Subscribe to IoT Core stream closed.");
        }
    }
}
```

Python (IPC client V1)

Example Esempio: sottoscrizione ai messaggi

> **Note**
>
> Questo esempio presuppone che si stia utilizzando la versione 1.5.4 o successiva di for SDK per dispositivi AWS IoT Python v2.

```
import time
import traceback

import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import (
    IoTCoreMessage,
    QOS,
    SubscribeToIoTCoreRequest
)

TIMEOUT = 10

ipc_client = awsiot.greengrasscoreipc.connect()

class StreamHandler(client.SubscribeToIoTCoreStreamHandler):
    def __init__(self):
        super().__init__()

    def on_stream_event(self, event: IoTCoreMessage) -> None:
        try:
```

```
            message = str(event.message.payload, "utf-8")
            topic_name = event.message.topic_name
            # Handle message.
        except:
            traceback.print_exc()

    def on_stream_error(self, error: Exception) -> bool:
        # Handle error.
        return True  # Return True to close stream, False to keep stream open.

    def on_stream_closed(self) -> None:
        # Handle close.
        pass


topic = "my/topic"
qos = QOS.AT_MOST_ONCE

request = SubscribeToIoTCoreRequest()
request.topic_name = topic
request.qos = qos
handler = StreamHandler()
operation = ipc_client.new_subscribe_to_iot_core(handler)
operation.activate(request)
future_response = operation.get_response()
future_response.result(TIMEOUT)

# Keep the main thread alive, or the process will exit.
while True:
    time.sleep(10)

# To stop subscribing, close the operation stream.
operation.close()
```

C++

Example Esempio: iscriversi ai messaggi

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
```

```
using namespace Aws::Greengrass;

class IoTCoreResponseHandler : public SubscribeToIoTCoreStreamHandler {

    public:
        virtual ~IoTCoreResponseHandler() {}

    private:
        void OnStreamEvent(IoTCoreMessage *response) override {
            auto message = response->GetMessage();
            if (message.has_value() && message.value().GetPayload().has_value()) {
                auto messageBytes = message.value().GetPayload().value();
                std::string messageString(messageBytes.begin(), messageBytes.end());
                std::string topicName =
 message.value().GetTopicName().value().c_str();
                // Handle message.
            }
        }

        bool OnStreamError(OperationError *error) override {
            // Handle error.
            return false; // Return true to close stream, false to keep stream open.
        }

        void OnStreamClosed() override {
            // Handle close.
        }
};

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        // Handle connection to IPC service.
    }

    void OnDisconnectCallback(RpcError error) override {
        // Handle disconnection from IPC service.
    }

    bool OnErrorCallback(RpcError error) override {
        // Handle IPC service connection error.
        return true;
    }
};
```

```
int main() {
    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    String topic("my/topic");
    QOS qos = QOS_AT_MOST_ONCE;
    int timeout = 10;

    SubscribeToIoTCoreRequest request;
    request.SetTopicName(topic);
    request.SetQos(qos);
    auto streamHandler = MakeShared<IoTCoreResponseHandler>(DefaultAllocator());
    auto operation = ipcClient.NewSubscribeToIoTCore(streamHandler);
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
 Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (!response) {
        // Handle error.
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            (void)error;
            // Handle operation error.
        } else {
            // Handle RPC error.
```

```
        }
        exit(-1);
    }

    // Keep the main thread alive, or the process will exit.
    while (true) {
        std::this_thread::sleep_for(std::chrono::seconds(10));
    }

    operation->Close();
    return 0;
}
```

JavaScript

Example Esempio: iscriversi ai messaggi

```
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";
import {IoTCoreMessage, QOS, SubscribeToIoTCoreRequest} from "aws-iot-device-sdk-v2/
dist/greengrasscoreipc/model";
import {RpcError} from "aws-iot-device-sdk-v2/dist/eventstream_rpc";

class SubscribeToIoTCore {
    private ipcClient: greengrasscoreipc.Client
    private readonly topic: string;

    constructor() {
        // define your own constructor, e.g.
        this.topic = "<define_your_topic>";
        this.subscribeToIoTCore().then(r => console.log("Started workflow"));
    }

    private async subscribeToIoTCore() {
        try {
            const request: SubscribeToIoTCoreRequest = {
                topicName: this.topic,
                qos: QOS.AT_LEAST_ONCE, // you can change this depending on your use
 case
            }

            this.ipcClient = await getIpcClient();

            const streamingOperation = this.ipcClient.subscribeToIoTCore(request);
```

```
            streamingOperation.on('message', (message: IoTCoreMessage) => {
                // parse the message depending on your use cases, e.g.
                if (message.message && message.message.payload) {
                    const receivedMessage = message.message.payload.toString();
                }
            });

            streamingOperation.on('streamError', (error : RpcError) => {
                // define your own error handling logic
            });

            streamingOperation.on('ended', () => {
                // define your own logic
            });

            await streamingOperation.activate();

            // Keep the main thread alive, or the process will exit.
            await new Promise((resolve) => setTimeout(resolve, 10000))
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}

export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

// starting point
```

```
const subscribeToIoTCore = new SubscribeToIoTCore();
```

Esempi

Utilizzate i seguenti esempi per imparare a utilizzare il servizio AWS IoT Core MQTT IPC nei vostri componenti.

Esempio di editore AWS IoT Core MQTT (C++)

La seguente ricetta di esempio consente al componente di pubblicare su tutti gli argomenti.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.IoTCorePublisherCpp",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that publishes MQTT messages to IoT Core.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.mqttproxy": {
          "com.example.IoTCorePublisherCpp:mqttproxy:1": {
            "policyDescription": "Allows access to publish to all topics.",
            "operations": [
              "aws.greengrass#PublishToIoTCore"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "{artifacts:path}/greengrassv2_iotcore_publisher"
      },
      "Artifacts": [
```

```
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.IoTCorePublisherCpp/1.0.0/greengrassv2_iotcore_publisher",
          "Permission": {
            "Execute": "OWNER"
          }
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.IoTCorePublisherCpp
ComponentVersion: 1.0.0
ComponentDescription: A component that publishes MQTT messages to IoT Core.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.mqttproxy:
        com.example.IoTCorePublisherCpp:mqttproxy:1:
          policyDescription: Allows access to publish to all topics.
          operations:
            - aws.greengrass#PublishToIoTCore
          resources:
            - "*"
Manifests:
  - Lifecycle:
      run: "{artifacts:path}/greengrassv2_iotcore_publisher"
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.IoTCorePublisherCpp/1.0.0/greengrassv2_iotcore_publisher
        Permission:
          Execute: OWNER
```

L'applicazione C++ di esempio seguente mostra come utilizzare il servizio AWS IoT Core MQTT IPC su cui pubblicare messaggi. AWS IoT Core

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        std::cout << "OnConnectCallback" << std::endl;
    }

    void OnDisconnectCallback(RpcError error) override {
        std::cout << "OnDisconnectCallback: " << error.StatusToString() << std::endl;
        exit(-1);
    }

    bool OnErrorCallback(RpcError error) override {
        std::cout << "OnErrorCallback: " << error.StatusToString() << std::endl;
        return true;
    }
};

int main() {
    String message("Hello from the Greengrass IPC MQTT publisher (C++).");
    String topic("test/topic/cpp");
    QOS qos = QOS_AT_LEAST_ONCE;
    int timeout = 10;

    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }
```

```
    while (true) {
        PublishToIoTCoreRequest request;
        Vector<uint8_t> messageData({message.begin(), message.end()});
        request.SetTopicName(topic);
        request.SetPayload(messageData);
        request.SetQos(qos);

        auto operation = ipcClient.NewPublishToIoTCore();
        auto activate = operation->Activate(request, nullptr);
        activate.wait();

        auto responseFuture = operation->GetResult();
        if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
            std::cerr << "Operation timed out while waiting for response from
 Greengrass Core." << std::endl;
            exit(-1);
        }

        auto response = responseFuture.get();
        if (response) {
            std::cout << "Successfully published to topic: " << topic << std::endl;
        } else {
            // An error occurred.
            std::cout << "Failed to publish to topic: " << topic << std::endl;
            auto errorType = response.GetResultType();
            if (errorType == OPERATION_ERROR) {
                auto *error = response.GetOperationError();
                std::cout << "Operation error: " << error->GetMessage().value() <<
 std::endl;
            } else {
                std::cout << "RPC error: " << response.GetRpcError() << std::endl;
            }
            exit(-1);
        }

        std::this_thread::sleep_for(std::chrono::seconds(5));
    }

    return 0;
}
```

Esempio di abbonato AWS IoT Core MQTT (C++)

La seguente ricetta di esempio consente al componente di sottoscrivere tutti gli argomenti.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.IoTCoreSubscriberCpp",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "A component that subscribes to MQTT messages from IoT
 Core.",
  "ComponentPublisher": "Amazon",
  "ComponentConfiguration": {
    "DefaultConfiguration": {
      "accessControl": {
        "aws.greengrass.ipc.mqttproxy": {
          "com.example.IoTCoreSubscriberCpp:mqttproxy:1": {
            "policyDescription": "Allows access to subscribe to all topics.",
            "operations": [
              "aws.greengrass#SubscribeToIoTCore"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "{artifacts:path}/greengrassv2_iotcore_subscriber"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.IoTCoreSubscriberCpp/1.0.0/greengrassv2_iotcore_subscriber",
          "Permission": {
            "Execute": "OWNER"
          }
        }
      ]
```

```
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.IoTCoreSubscriberCpp
ComponentVersion: 1.0.0
ComponentDescription: A component that subscribes to MQTT messages from IoT Core.
ComponentPublisher: Amazon
ComponentConfiguration:
  DefaultConfiguration:
    accessControl:
      aws.greengrass.ipc.mqttproxy:
        com.example.IoTCoreSubscriberCpp:mqttproxy:1:
          policyDescription: Allows access to subscribe to all topics.
          operations:
            - aws.greengrass#SubscribeToIoTCore
          resources:
            - "*"
Manifests:
  - Lifecycle:
      run: "{artifacts:path}/greengrassv2_iotcore_subscriber"
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.IoTCoreSubscriberCpp/1.0.0/greengrassv2_iotcore_subscriber
        Permission:
          Execute: OWNER
```

L'applicazione C++ di esempio seguente mostra come utilizzare il servizio AWS IoT Core MQTT IPC per iscriversi ai messaggi da. AWS IoT Core

```
#include <iostream>

#include <aws/crt/Api.h>
#include <aws/greengrass/GreengrassCoreIpcClient.h>

using namespace Aws::Crt;
using namespace Aws::Greengrass;
```

```
class IoTCoreResponseHandler : public SubscribeToIoTCoreStreamHandler {

    public:
        virtual ~IoTCoreResponseHandler() {}

    private:

        void OnStreamEvent(IoTCoreMessage *response) override {
            auto message = response->GetMessage();
            if (message.has_value() && message.value().GetPayload().has_value()) {
                auto messageBytes = message.value().GetPayload().value();
                std::string messageString(messageBytes.begin(), messageBytes.end());
                std::string messageTopic =
 message.value().GetTopicName().value().c_str();
                std::cout << "Received new message on topic: " << messageTopic <<
 std::endl;
                std::cout << "Message: " << messageString << std::endl;
            }
        }

        bool OnStreamError(OperationError *error) override {
            std::cout << "Received an operation error: ";
            if (error->GetMessage().has_value()) {
                std::cout << error->GetMessage().value();
            }
            std::cout << std::endl;
            return false; // Return true to close stream, false to keep stream open.
        }

        void OnStreamClosed() override {
            std::cout << "Subscribe to IoT Core stream closed." << std::endl;
        }
};

class IpcClientLifecycleHandler : public ConnectionLifecycleHandler {
    void OnConnectCallback() override {
        std::cout << "OnConnectCallback" << std::endl;
    }

    void OnDisconnectCallback(RpcError error) override {
        std::cout << "OnDisconnectCallback: " << error.StatusToString() << std::endl;
        exit(-1);
    }
```

```
    bool OnErrorCallback(RpcError error) override {
        std::cout << "OnErrorCallback: " << error.StatusToString() << std::endl;
        return true;
    }
};

int main() {
    String topic("test/topic/cpp");
    QOS qos = QOS_AT_LEAST_ONCE;
    int timeout = 10;

    ApiHandle apiHandle(g_allocator);
    Io::EventLoopGroup eventLoopGroup(1);
    Io::DefaultHostResolver socketResolver(eventLoopGroup, 64, 30);
    Io::ClientBootstrap bootstrap(eventLoopGroup, socketResolver);
    IpcClientLifecycleHandler ipcLifecycleHandler;
    GreengrassCoreIpcClient ipcClient(bootstrap);
    auto connectionStatus = ipcClient.Connect(ipcLifecycleHandler).get();
    if (!connectionStatus) {
        std::cerr << "Failed to establish IPC connection: " <<
 connectionStatus.StatusToString() << std::endl;
        exit(-1);
    }

    SubscribeToIoTCoreRequest request;
    request.SetTopicName(topic);
    request.SetQos(qos);
    auto streamHandler = MakeShared<IoTCoreResponseHandler>(DefaultAllocator());
    auto operation = ipcClient.NewSubscribeToIoTCore(streamHandler);
    auto activate = operation->Activate(request, nullptr);
    activate.wait();

    auto responseFuture = operation->GetResult();
    if (responseFuture.wait_for(std::chrono::seconds(timeout)) ==
 std::future_status::timeout) {
        std::cerr << "Operation timed out while waiting for response from Greengrass
 Core." << std::endl;
        exit(-1);
    }

    auto response = responseFuture.get();
    if (response) {
        std::cout << "Successfully subscribed to topic: " << topic << std::endl;
    } else {
```

```
        // An error occurred.
        std::cout << "Failed to subscribe to topic: " << topic << std::endl;
        auto errorType = response.GetResultType();
        if (errorType == OPERATION_ERROR) {
            auto *error = response.GetOperationError();
            std::cout << "Operation error: " << error->GetMessage().value() <<
 std::endl;
        } else {
            std::cout << "RPC error: " << response.GetRpcError() << std::endl;
        }
        exit(-1);
    }

    // Keep the main thread alive, or the process will exit.
    while (true) {
        std::this_thread::sleep_for(std::chrono::seconds(10));
    }

    operation->Close();
    return 0;
}
```

Interagisci con il ciclo di vita dei componenti

Utilizza il servizio IPC del ciclo di vita dei componenti per:

- Aggiorna lo stato del componente sul dispositivo principale.
- Iscriviti agli aggiornamenti sullo stato dei componenti.
- Impedisci al nucleus di interrompere il componente per applicare un aggiornamento durante una distribuzione.
- Metti in pausa e riprendi i processi dei componenti.

Argomenti

- Versioni SDK minime
- Autorizzazione
- UpdateState
- SubscribeToComponentUpdates
- DeferComponentUpdate

- PauseComponent
- ResumeComponent

Versioni SDK minime

La tabella seguente elenca le versioni minime da utilizzare per interagire con il SDK per dispositivi AWS IoT ciclo di vita dei componenti.

SDK	Versione minima	
SDK per dispositivi AWS IoTper Java v2	v1.2.10	
SDK per dispositivi AWS IoTper Python v2	v1.5.3	
SDK per dispositivi AWS IoTper C++ v2	v1.17.0	
SDK per dispositivi AWS IoTper v2 JavaScript	v1.12.0	

Autorizzazione

Per mettere in pausa o riprendere altri componenti di un componente personalizzato, è necessario definire politiche di autorizzazione che consentano al componente di gestire altri componenti. Per informazioni sulla definizione delle politiche di autorizzazione, vedere. Autorizza i componenti a eseguire operazioni IPC

Le politiche di autorizzazione per la gestione del ciclo di vita dei componenti hanno le seguenti proprietà.

Identificatore del servizio IPC: `aws.greengrass.ipc.lifecycle`

Operazione	Descrizione	Risorse
`aws.greengrass#PauseComponent`	Consente a un componente di mettere in pausa i componenti specificati.	Un nome di componente o * per consentire l'accesso a tutti i componenti.
`aws.greengrass#ResumeComponent`	Consente a un componente di ripristinare i componenti specificati.	Un nome di componente o * per consentire l'accesso a tutti i componenti.
*	Consente a un componente di mettere in pausa e riprendere i componenti specificati.	Un nome di componente o * per consentire l'accesso a tutti i componenti.

Esempi di politiche di autorizzazione

È possibile fare riferimento al seguente esempio di politica di autorizzazione per configurare le politiche di autorizzazione per i componenti.

Example Esempio di politica di autorizzazione

Il seguente esempio di politica di autorizzazione consente a un componente di mettere in pausa e riprendere tutti i componenti.

```
{
  "accessControl": {
    "aws.greengrass.ipc.lifecycle": {
      "com.example.MyLocalLifecycleComponent:lifecycle:1": {
        "policyDescription": "Allows access to pause/resume all components.",
        "operations": [
          "aws.greengrass#PauseComponent",
          "aws.greengrass#ResumeComponent"
        ],
        "resources": [
          "*"
        ]
      }
    }
  }
}
```

UpdateState

Aggiorna lo stato del componente sul dispositivo principale.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`state`

Lo stato da impostare. Questo enum ha `LifecycleState` i seguenti valori:

- `RUNNING`
- `ERRORED`

Risposta

Questa operazione non fornisce alcuna informazione nella sua risposta.

SubscribeToComponentUpdates

Iscriviti per ricevere notifiche prima che il software AWS IoT Greengrass Core aggiorni un componente. La notifica specifica se il nucleo verrà riavviato o meno come parte dell'aggiornamento.

Il nucleus invia notifiche di aggiornamento solo se la politica di aggiornamento dei componenti della distribuzione specifica di notificare i componenti. Il comportamento predefinito consiste nel notificare i componenti. Per ulteriori informazioni, vedete Creare distribuzione l'DeploymentComponentUpdatePolicyoggetto che potete fornire quando chiamate l'CreateDeploymentoperazione.

> **Important**
>
> Le distribuzioni locali non notificano i componenti prima degli aggiornamenti.

Questa operazione è un'operazione di sottoscrizione in cui ci si iscrive a un flusso di messaggi di eventi. Per utilizzare questa operazione, definite un gestore di risposte di flusso con funzioni che gestiscono i messaggi di evento, gli errori e la chiusura dei flussi. Per ulteriori informazioni, consulta Iscriviti ai flussi di eventi IPC.

Tipo di messaggio di evento: `ComponentUpdatePolicyEvents`

> **Tip**
>
> Puoi seguire un tutorial per imparare a sviluppare un componente che rinvii in modo condizionale gli aggiornamenti dei componenti. Per ulteriori informazioni, consulta Tutorial: Sviluppa un componente Greengrass che rinvii gli aggiornamenti dei componenti.

Richiesta

La richiesta di questa operazione non ha parametri.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`messages`

Il flusso di messaggi di notifica. Questo oggetto contiene `ComponentUpdatePolicyEvents` le seguenti informazioni:

`preUpdateEvent`(Python:) `pre_update_event`

(Facoltativo) Un evento che indica che il nucleo desidera aggiornare un componente. È possibile rispondere con l'DeferComponentUpdateoperazione di conferma o rinviare l'aggiornamento fino a quando il componente non sarà pronto per il riavvio. Questo oggetto contiene `PreComponentUpdateEvent` le seguenti informazioni:

`deploymentId`(Python:) `deployment_id`

L'ID della AWS IoT Greengrass distribuzione che aggiorna il componente.

`isGgcRestarting`(Python:) `is_ggc_restarting`

Se il nucleo deve essere riavviato o meno per applicare l'aggiornamento.

`postUpdateEvent`(Python:) `post_update_event`

(Facoltativo) Un evento che indica che il nucleo ha aggiornato un componente. Questo oggetto contiene `PostComponentUpdateEvent` le seguenti informazioni:

`deploymentId`(Python:) `deployment_id`

L'ID della AWS IoT Greengrass distribuzione che ha aggiornato il componente.

> **Note**
>
> Questa funzionalità richiede la versione 2.7.0 o successiva del componente Greengrass nucleus.

DeferComponentUpdate

Riconosci o rimanda l'aggiornamento di un componente con cui lo scopri. SubscribeToComponentUpdates Specificate il tempo di attesa prima che il nucleo verifichi nuovamente se il componente è pronto per procedere con l'aggiornamento del componente. È inoltre possibile utilizzare questa operazione per comunicare al nucleo che il componente è pronto per l'aggiornamento.

Se un componente non risponde alla notifica di aggiornamento del componente, il nucleus attende il periodo di tempo specificato nella politica di aggiornamento dei componenti della distribuzione. Dopo tale timeout, il nucleus procede con la distribuzione. Il timeout predefinito per l'aggiornamento dei componenti è di 60 secondi. Per ulteriori informazioni, vedere Creare distribuzione e l'DeploymentComponentUpdatePolicyoggetto che è possibile fornire quando si chiama l'CreateDeploymentoperazione.

> **Tip**
>
> Puoi seguire un tutorial per imparare a sviluppare un componente che differisca in modo condizionale gli aggiornamenti dei componenti. Per ulteriori informazioni, consulta Tutorial: Sviluppa un componente Greengrass che rinvii gli aggiornamenti dei componenti.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`deploymentId`(Python:) `deployment_id`

L'ID della AWS IoT Greengrass distribuzione da rinviare.

`message`

(Facoltativo) Il nome del componente per il quale posticipare gli aggiornamenti.

Il valore predefinito è il nome del componente che effettua la richiesta.

`recheckAfterMs`(Python:) `recheck_after_ms`

Il periodo di tempo in millisecondi per il quale posticipare l'aggiornamento. Il nucleo attende questo intervallo di tempo e poi ne invia un altro `PreComponentUpdateEvent` con cui è possibile eseguire l'individuazione. SubscribeToComponentUpdates

`0`Specificare di confermare l'aggiornamento. Questo indica al nucleo che il componente è pronto per l'aggiornamento.

Il valore predefinito è zero millisecondi, il che significa che conferma l'aggiornamento.

Risposta

Questa operazione non fornisce alcuna informazione nella sua risposta.

PauseComponent

Questa funzionalità è disponibile per la versione 2.4.0 e successive del componente Greengrass nucleus. AWS IoT Greengrassattualmente non supporta questa funzionalità sui dispositivi Windows core.

Sospende i processi di un componente sul dispositivo principale. Per riprendere un componente, utilizzate l'ResumeComponentoperazione.

È possibile mettere in pausa solo i componenti generici. Se si tenta di mettere in pausa qualsiasi altro tipo di componente, questa operazione genera un. `InvalidRequestError`

> **Note**
>
> Questa operazione non può mettere in pausa i processi containerizzati, come i contenitori Docker. Per mettere in pausa e riprendere un contenitore Docker, puoi utilizzare i comandi docker pause e docker unpause.

Questa operazione non mette in pausa le dipendenze dei componenti o i componenti che dipendono dal componente che metti in pausa. Considerate questo comportamento quando mettete in pausa un componente che dipende da un altro componente, poiché il componente dipendente potrebbe riscontrare problemi quando la sua dipendenza viene messa in pausa.

Quando si riavvia o si spegne un componente in pausa, ad esempio durante una distribuzione, il Greengrass nucleus riprende il componente ed esegue il suo ciclo di vita di spegnimento. Per ulteriori informazioni sul riavvio di un componente, vedere. RestartComponent

> **Important**
>
> Per utilizzare questa operazione, è necessario definire una politica di autorizzazione che conceda il permesso di utilizzare questa operazione. Per ulteriori informazioni, consulta Autorizzazione.

Versioni SDK minime

La tabella seguente elenca le versioni minime da utilizzare per mettere in pausa e riprendere i componenti. SDK per dispositivi AWS IoT

SDK	Versione minima
SDK per dispositivi AWS IoTper Java v2	v1.4.3
SDK per dispositivi AWS IoTper Python v2	v1.6.2
SDK per dispositivi AWS IoTper C++ v2	v1.13.1
SDK per dispositivi AWS IoTper v2 JavaScript	v1.12.0

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`componentName`(Python:) `component_name`

Il nome del componente da mettere in pausa, che deve essere un componente generico. Per ulteriori informazioni, consulta Tipi di componenti.

Risposta

Questa operazione non fornisce alcuna informazione nella sua risposta.

ResumeComponent

Questa funzionalità è disponibile per la versione 2.4.0 e successive del componente Greengrass nucleus. AWS IoT Greengrassattualmente non supporta questa funzionalità sui dispositivi Windows core.

Riprende i processi di un componente sul dispositivo principale. Per mettere in pausa un componente, utilizzate l'operazione. PauseComponent

È possibile riprendere solo i componenti in pausa. Se si tenta di riprendere un componente che non è in pausa, questa operazione genera un. `InvalidRequestError`

> **Important**
>
> Per utilizzare questa operazione, è necessario definire una politica di autorizzazione che conceda l'autorizzazione a farlo. Per ulteriori informazioni, consulta Autorizzazione.

Versioni SDK minime

La tabella seguente elenca le versioni minime da utilizzare per mettere in pausa e riprendere i componenti. SDK per dispositivi AWS IoT

SDK	Versione minima	
SDK per dispositivi AWS IoTper Java v2	v1.4.3	
SDK per dispositivi AWS IoTper Python v2	v1.6.2	
SDK per dispositivi AWS IoTper C++ v2	v1.13.1	
SDK per dispositivi AWS IoTper v2 JavaScript	v1.12.0	

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`componentName`(Python:) `component_name`

Il nome del componente da riprendere.

Risposta

Questa operazione non fornisce alcuna informazione nella sua risposta.

Interazione con la configurazione dei componenti

Il servizio IPC di configurazione dei componenti consente di effettuare le seguenti operazioni:

- Ottieni e imposta i parametri di configurazione dei componenti.
- Iscriviti agli aggiornamenti della configurazione dei componenti.
- Convalida gli aggiornamenti della configurazione dei componenti prima che il nucleo li applichi.

Argomenti

Versioni SDK minime

La tabella seguente elenca le versioni minime di SDK per dispositivi AWS IoT da utilizzare per interagire con la configurazione dei componenti.

SDK	Versione minima	
SDK per dispositivi AWS IoTper Java v2	v1.2.10	
SDK per dispositivi AWS IoTper Python v2	v1.5.3	
SDK per dispositivi AWS IoTper C++ v2	v1.17.0	
SDK per dispositivi AWS IoTper v2 JavaScript	v1.12.0	

GetConfiguration

Ottiene un valore di configurazione per un componente sul dispositivo principale. Specificate il percorso chiave per il quale ottenere un valore di configurazione.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`componentName`(Python:) `component_name`

(Facoltativo) Il nome del componente.

Il valore predefinito è il nome del componente che effettua la richiesta.

`keyPath`(Python:) `key_path`

Il percorso chiave verso il valore di configurazione. Specificate un elenco in cui ogni voce è la chiave per un singolo livello nell'oggetto di configurazione. Ad esempio, specificare `["mqtt", "port"]` di ottenere il valore di `port` nella seguente configurazione.

```
{
  "mqtt": {
    "port": 443
  }
}
```

Per ottenere la configurazione completa del componente, specificate un elenco vuoto.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`componentName`(Python:) `component_name`

Il nome del componente.

`value`

La configurazione richiesta come oggetto.

UpdateConfiguration

Aggiorna un valore di configurazione per questo componente sul dispositivo principale.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`keyPath`(Python:) `key_path`

(Facoltativo) Il percorso chiave del nodo contenitore (l'oggetto) da aggiornare. Specificate un elenco in cui ogni voce è la chiave per un singolo livello nell'oggetto di configurazione. Ad esempio, specificate il percorso della chiave `["mqtt"]` e il valore `{ "port": 443 }` di unione di cui impostare il valore `port` nella seguente configurazione.

```
{
  "mqtt": {
    "port": 443
  }
}
```

Il percorso chiave deve specificare un nodo contenitore (un oggetto) nella configurazione. Se il nodo non esiste nella configurazione del componente, questa operazione lo crea e imposta il suo valore sull'oggetto in`valueToMerge`.

Il valore predefinito è la radice dell'oggetto di configurazione.

`timestamp`

L'ora attuale dell'epoca Unix in millisecondi. Questa operazione utilizza questo timestamp per risolvere gli aggiornamenti simultanei della chiave. Se la chiave nella configurazione del componente ha un timestamp maggiore di quello della richiesta, la richiesta ha esito negativo.

`valueToMerge`(Python:) `value_to_merge`

L'oggetto di configurazione da unire nella posizione specificata. `keyPath` Per ulteriori informazioni, consulta Aggiornamento delle configurazioni dei componenti.

Risposta

Questa operazione non fornisce alcuna informazione nella sua risposta.

SubscribeToConfigurationUpdate

Iscriviti per ricevere notifiche quando la configurazione di un componente viene aggiornata. Quando sottoscrivi una chiave, ricevi una notifica ogni volta che un elemento secondario di quella chiave si aggiorna.

Questa operazione è un'operazione di sottoscrizione in cui ti iscrivi a un flusso di messaggi di eventi. Per utilizzare questa operazione, definite un gestore di risposte di flusso con funzioni che gestiscono i messaggi di evento, gli errori e la chiusura dei flussi. Per ulteriori informazioni, consulta Iscriviti ai flussi di eventi IPC.

Tipo di messaggio di evento: `ConfigurationUpdateEvents`

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`componentName`(Python:) `component_name`

(Facoltativo) Il nome del componente.

Il valore predefinito è il nome del componente che effettua la richiesta.

`keyPath`(Python:) `key_path`

Il percorso chiave del valore di configurazione a cui sottoscrivere. Specificate un elenco in cui ogni voce è la chiave per un singolo livello nell'oggetto di configurazione. Ad esempio, specificare `["mqtt", "port"]` di ottenere il valore di `port` nella seguente configurazione.

```
{
  "mqtt": {
    "port": 443
  }
}
```

Per sottoscrivere gli aggiornamenti per tutti i valori nella configurazione del componente, specificate un elenco vuoto.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`messages`

Il flusso di messaggi di notifica. Questo oggetto contiene `ConfigurationUpdateEvents` le seguenti informazioni:

`configurationUpdateEvent`(Python:) `configuration_update_event`

L'evento di aggiornamento della configurazione. Questo oggetto contiene `ConfigurationUpdateEvent` le seguenti informazioni:

`componentName`(Python:) `component_name`

Il nome del componente.

`keyPath`(Python:) `key_path`

Il percorso chiave del valore di configurazione che è stato aggiornato.

SubscribeToValidateConfigurationUpdates

Iscriviti per ricevere notifiche prima degli aggiornamenti della configurazione di questo componente. Ciò consente ai componenti di convalidare gli aggiornamenti alla propria configurazione. Utilizzate l'SendConfigurationValidityReportoperazione per indicare al nucleo se la configurazione è valida o meno.

> **Important**
>
> Le distribuzioni locali non notificano ai componenti gli aggiornamenti.

Questa operazione è un'operazione di sottoscrizione in cui ci si iscrive a un flusso di messaggi di eventi. Per utilizzare questa operazione, definite un gestore di risposte di flusso con funzioni che gestiscono i messaggi di evento, gli errori e la chiusura dei flussi. Per ulteriori informazioni, consulta Iscriviti ai flussi di eventi IPC.

Tipo di messaggio di evento: `ValidateConfigurationUpdateEvents`

Richiesta

La richiesta di questa operazione non ha parametri.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`messages`

Il flusso di messaggi di notifica. Questo oggetto contiene `ValidateConfigurationUpdateEvents` le seguenti informazioni:

`validateConfigurationUpdateEvent`(Python:) `validate_configuration_update_event`

L'evento di aggiornamento della configurazione. Questo oggetto contiene `ValidateConfigurationUpdateEvent` le seguenti informazioni:

`deploymentId`(Python:) `deployment_id`

L'ID della AWS IoT Greengrass distribuzione che aggiorna il componente.

`configuration`

L'oggetto che contiene la nuova configurazione.

SendConfigurationValidityReport

Indica al nucleo se un aggiornamento della configurazione di questo componente è valido o meno. L'implementazione fallisce se dici al nucleus che la nuova configurazione non è valida. Utilizza l'SubscribeToValidateConfigurationUpdatesoperazione per sottoscrivere e convalidare gli aggiornamenti di configurazione.

Se un componente non risponde a una notifica di convalida dell'aggiornamento della configurazione, il nucleus attende il periodo di tempo specificato nella politica di convalida della configurazione

della distribuzione. Dopo tale timeout, il nucleo procede con la distribuzione. Il timeout di convalida dei componenti predefinito è di 20 secondi. Per ulteriori informazioni, vedere Creare distribuzione e l'DeploymentConfigurationValidationPolicyoggetto che è possibile fornire quando si chiama l'CreateDeploymentoperazione.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`configurationValidityReport`(Python:) `configuration_validity_report`

Il rapporto che indica al nucleo se l'aggiornamento della configurazione è valido o meno. Questo oggetto contiene `ConfigurationValidityReport` le seguenti informazioni:

`status`

Lo stato di validità. Questo enum`ConfigurationValidityStatus`, ha i seguenti valori:

- `ACCEPTED`— La configurazione è valida e il nucleo può applicarla a questo componente.
- `REJECTED`— La configurazione non è valida e l'implementazione non riesce.

`deploymentId`(Python:) `deployment_id`

L'ID della AWS IoT Greengrass distribuzione che ha richiesto l'aggiornamento della configurazione.

`message`

(Facoltativo) Un messaggio che riporta il motivo per cui la configurazione non è valida.

Risposta

Questa operazione non fornisce alcuna informazione nella sua risposta.

Recupera valori segreti

Utilizza il servizio IPC di gestione dei segreti per recuperare i valori segreti dai segreti sul dispositivo principale. Utilizzi il componente secret manager per distribuire segreti crittografati sui dispositivi principali. Quindi, puoi utilizzare un'operazione IPC per decrittografare il segreto e utilizzarne il valore nei componenti personalizzati.

Argomenti

Versioni SDK minime

La tabella seguente elenca le versioni minime di da utilizzare per recuperare i valori segreti dai segreti sul dispositivo principale. SDK per dispositivi AWS IoT

SDK	Versione minima	
SDK per dispositivi AWS IoTper Java v2	v1.2.10	
SDK per dispositivi AWS IoTper Python v2	v1.5.3	
SDK per dispositivi AWS IoTper C++ v2	v1.17.0	
SDK per dispositivi AWS IoTper v2 JavaScript	v1.12.0	

Autorizzazione

Per utilizzare il gestore segreto in un componente personalizzato, è necessario definire politiche di autorizzazione che consentano al componente di ottenere il valore dei segreti archiviati sul dispositivo principale. Per informazioni sulla definizione delle politiche di autorizzazione, vedereAutorizza i componenti a eseguire operazioni IPC.

Le politiche di autorizzazione per il gestore segreto hanno le seguenti proprietà.

Identificatore del servizio IPC: `aws.greengrass.SecretManager`

Operazione	Descrizione	Risorse
`aws.greengrass#GetSecretValue`	Consente a un componente di ottenere il valore dei segreti crittografati sul dispositivo principale.	Un ARN segreto di Secrets Manager o * per consentire l'accesso a tutti i segreti.

Esempi di politiche di autorizzazione

È possibile fare riferimento al seguente esempio di politica di autorizzazione per configurare le politiche di autorizzazione per i componenti.

Example Esempio di politica di autorizzazione

Il seguente esempio di politica di autorizzazione consente a un componente di ottenere il valore di qualsiasi segreto sul dispositivo principale.

> **Note**
>
> In un ambiente di produzione, si consiglia di ridurre l'ambito della politica di autorizzazione, in modo che il componente recuperi solo i segreti che utilizza. È possibile modificare il carattere * jolly in un elenco di ARN segreti quando si distribuisce il componente.

```
{
  "accessControl": {
    "aws.greengrass.SecretManager": {
      "com.example.MySecretComponent:secrets:1": {
        "policyDescription": "Allows access to a secret.",
        "operations": [
          "aws.greengrass#GetSecretValue"
        ],
        "resources": [
          "*"
        ]
      }
    }
  }
}
```

GetSecretValue

Ottiene il valore di un segreto archiviato nel dispositivo principale.

Questa operazione è simile all'operazione Secrets Manager che è possibile utilizzare per ottenere il valore di un segreto inCloud AWS. Per ulteriori informazioni, consulta GetSecretValue nella documentazione di riferimento dell'API AWS Secrets Manager.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`secretId`(Python:) `secret_id`

Il nome del segreto da ottenere. Puoi specificare l'Amazon Resource Name (ARN) o il nome descrittivo del segreto.

`versionId`(Python:) `version_id`

(Facoltativo) L'ID della versione da ottenere.

È possibile specificare `versionId` o `versionStage`.

Se non specifichi `versionId` o`versionStage`, per impostazione predefinita per questa operazione viene utilizzata la versione con l'`AWSCURRENT`etichetta.

`versionStage`(Python:) `version_stage`

(Facoltativo) L'etichetta temporanea della versione da scaricare.

È possibile specificare `versionId` o `versionStage`.

Se non specifichi `versionId` o`versionStage`, per impostazione predefinita per questa operazione viene utilizzata la versione con l'etichetta. `AWSCURRENT`

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`secretId`(Python:) `secret_id`

L'ID del segreto.

`versionId`(Python:) `version_id`

L'ID di questa versione del segreto.

`versionStage`(Python:) `version_stage`

L'elenco delle etichette di staging allegate a questa versione del segreto.

`secretValue`(Python:) `secret_value`

Il valore di questa versione del segreto. Questo oggetto contiene `SecretValue` le seguenti informazioni.

`secretString`(Python:) `secret_string`

La parte decrittografata delle informazioni segrete protette che hai fornito a Secrets Manager come stringa.

`secretBinary`(Python:) `secret_binary`

(Facoltativo) La parte decrittografata delle informazioni segrete protette fornite a Secrets Manager come dati binari sotto forma di matrice di byte. Questa proprietà contiene i dati binari sotto forma di stringa con codifica Base64.

Questa proprietà non viene utilizzata se hai creato il segreto nella console Secrets Manager.

Esempi

Gli esempi seguenti mostrano come chiamare questa operazione nel codice componente personalizzato.

Java (IPC client V1)

Example Esempio: ottieni un valore segreto

> **Note**
>
> Questo esempio utilizza una `IPCUtils` classe per creare una connessione al servizio AWS IoT Greengrass Core IPC. Per ulteriori informazioni, consulta Connect al servizio AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;
```

```
import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GetSecretValueResponseHandler;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.model.GetSecretValueRequest;
import software.amazon.awssdk.aws.greengrass.model.GetSecretValueResponse;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class GetSecretValue {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        String secretArn = args[0];
        String versionStage = args[1];
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            GetSecretValueResponseHandler responseHandler =
                    GetSecretValue.getSecretValue(ipcClient, secretArn,
 versionStage);
            CompletableFuture<GetSecretValueResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                GetSecretValueResponse response =
 futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                response.getSecretValue().postFromJson();
                String secretString = response.getSecretValue().getSecretString();
                System.out.println("Successfully retrieved secret value: " +
 secretString);
            } catch (TimeoutException e) {
                System.err.println("Timeout occurred while retrieving secret: " +
 secretArn);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.println("Unauthorized error while retrieving secret:
 " + secretArn);
```

```
                } else {
                    throw e;
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static GetSecretValueResponseHandler
 getSecretValue(GreengrassCoreIPCClient greengrassCoreIPCClient, String secretArn,
 String versionStage) {
        GetSecretValueRequest getSecretValueRequest = new GetSecretValueRequest();
        getSecretValueRequest.setSecretId(secretArn);
        getSecretValueRequest.setVersionStage(versionStage);
        return greengrassCoreIPCClient.getSecretValue(getSecretValueRequest,
 Optional.empty());
    }
}
```

Python (IPC client V1)

Example Esempio: ottieni un valore segreto

> **Note**
>
> Questo esempio presuppone che si stia utilizzando la versione 1.5.4 o successiva di for SDK per dispositivi AWS IoT Python v2.

```
import json

import awsiot.greengrasscoreipc
from awsiot.greengrasscoreipc.model import (
    GetSecretValueRequest,
    GetSecretValueResponse,
    UnauthorizedError
)
```

```
secret_id = 'arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyGreengrassSecret-abcdef'
TIMEOUT = 10

ipc_client = awsiot.greengrasscoreipc.connect()

request = GetSecretValueRequest()
request.secret_id = secret_id
request.version_stage = 'AWSCURRENT'
operation = ipc_client.new_get_secret_value()
operation.activate(request)
future_response = operation.get_response()
response = future_response.result(TIMEOUT)
secret_json = json.loads(response.secret_value.secret_string)
# Handle secret value.
```

JavaScript

Example Esempio: ottieni un valore segreto

```
import {
    GetSecretValueRequest,
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from "aws-iot-device-sdk-v2/dist/greengrasscoreipc";

class GetSecretValue {
    private readonly secretId : string;
    private readonly versionStage : string;
    private ipcClient : greengrasscoreipc.Client

    constructor() {
        this.secretId = "<define_your_own_secretId>"
        this.versionStage = "<define_your_own_versionStage>"

        this.getSecretValue().then(r => console.log("Started workflow"));
    }

    private async getSecretValue() {
        try {
            this.ipcClient = await getIpcClient();
```

```
            const getSecretValueRequest : GetSecretValueRequest = {
                secretId: this.secretId,
                versionStage: this.versionStage,
            };

            const result = await
 this.ipcClient.getSecretValue(getSecretValueRequest);
            const secretString = result.secretValue.secretString;
            console.log("Successfully retrieved secret value: " + secretString)
        } catch (e) {
            // parse the error depending on your use cases
            throw e
        }
    }
}

export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

const getSecretValue = new GetSecretValue();
```

Esempi

Utilizza i seguenti esempi per imparare a utilizzare il servizio IPC di gestione segreta nei tuoi componenti.

Esempio: segreto di stampa (Python, client IPC V1)

Questo componente di esempio stampa il valore di un segreto che viene distribuito sul dispositivo principale.

> **Important**
>
> Questo componente di esempio stampa il valore di un segreto, quindi usalo solo con i segreti che memorizzano i dati di test. Non utilizzate questo componente per stampare il valore di un segreto che memorizza informazioni importanti.

Argomenti

Recipe

La seguente ricetta di esempio definisce un parametro di configurazione ARN segreto e consente al componente di ottenere il valore di qualsiasi segreto sul dispositivo principale.

> **Note**
>
> In un ambiente di produzione, si consiglia di ridurre l'ambito della politica di autorizzazione, in modo che il componente recuperi solo i segreti che utilizza. È possibile modificare il carattere * jolly in un elenco di ARN segreti quando si distribuisce il componente.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.PrintSecret",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Prints the value of an AWS Secrets Manager secret.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.SecretManager": {
      "VersionRequirement": "^2.0.0",
      "DependencyType": "HARD"
    }
  },
  "ComponentConfiguration": {
```

```
    "DefaultConfiguration": {
      "SecretArn": "",
      "accessControl": {
        "aws.greengrass.SecretManager": {
          "com.example.PrintSecret:secrets:1": {
            "policyDescription": "Allows access to a secret.",
            "operations": [
              "aws.greengrass#GetSecretValue"
            ],
            "resources": [
              "*"
            ]
          }
        }
      }
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "python3 -m pip install --user awsiotsdk",
        "run": "python3 -u {artifacts:path}/print_secret.py \"{configuration:/
SecretArn}\""
      }
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "py -3 -m pip install --user awsiotsdk",
        "run": "py -3 -u {artifacts:path}/print_secret.py \"{configuration:/
SecretArn}\""
      }
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.PrintSecret
ComponentVersion: 1.0.0
ComponentDescription: Prints the value of a Secrets Manager secret.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.SecretManager:
    VersionRequirement: "^2.0.0"
    DependencyType: HARD
ComponentConfiguration:
  DefaultConfiguration:
    SecretArn: ''
    accessControl:
      aws.greengrass.SecretManager:
        com.example.PrintSecret:secrets:1:
          policyDescription: Allows access to a secret.
          operations:
            - aws.greengrass#GetSecretValue
          resources:
            - "*"
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: python3 -m pip install --user awsiotsdk
      run: python3 -u {artifacts:path}/print_secret.py "{configuration:/SecretArn}"
  - Platform:
      os: windows
    Lifecycle:
      install: py -3 -m pip install --user awsiotsdk
      run: py -3 -u {artifacts:path}/print_secret.py "{configuration:/SecretArn}"
```

Artifacts

L'esempio seguente di applicazione Python dimostra come utilizzare il servizio IPC del gestore segreto per ottenere il valore di un segreto sul dispositivo principale.

```
import concurrent.futures
import json
```

```
import sys
import traceback

import awsiot.greengrasscoreipc
from awsiot.greengrasscoreipc.model import (
    GetSecretValueRequest,
    GetSecretValueResponse,
    UnauthorizedError
)

TIMEOUT = 10

if len(sys.argv) == 1:
    print('Provide SecretArn in the component configuration.', file=sys.stdout)
    exit(1)

secret_id = sys.argv[1]

try:
    ipc_client = awsiot.greengrasscoreipc.connect()

    request = GetSecretValueRequest()
    request.secret_id = secret_id
    operation = ipc_client.new_get_secret_value()
    operation.activate(request)
    future_response = operation.get_response()

    try:
        response = future_response.result(TIMEOUT)
        secret_json = json.loads(response.secret_value.secret_string)
        print('Successfully got secret: ' + secret_id)
        print('Secret value: ' + str(secret_json))
    except concurrent.futures.TimeoutError:
        print('Timeout occurred while getting secret: ' + secret_id, file=sys.stderr)
    except UnauthorizedError as e:
        print('Unauthorized error while getting secret: ' + secret_id,
 file=sys.stderr)
        raise e
    except Exception as e:
        print('Exception while getting secret: ' + secret_id, file=sys.stderr)
        raise e
except Exception:
    print('Exception occurred when using IPC.', file=sys.stderr)
    traceback.print_exc()
```

```
    exit(1)
```

Utilizzo

È possibile utilizzare questo componente di esempio con il componente secret manager per distribuire e stampare il valore di un segreto sul dispositivo principale.

Per creare, distribuire e stampare un segreto di test

1. Crea un segreto di Secrets Manager con i dati di test.

   Linux or Unix

   ```
   aws secretsmanager create-secret \
     --name MyTestGreengrassSecret \
     --secret-string '{"my-secret-key": "my-secret-value"}'
   ```

   Windows Command Prompt (CMD)

   ```
   aws secretsmanager create-secret ^
     --name MyTestGreengrassSecret ^
     --secret-string '{"my-secret-key": "my-secret-value"}'
   ```

   PowerShell

   ```
   aws secretsmanager create-secret `
     --name MyTestGreengrassSecret `
     --secret-string '{"my-secret-key": "my-secret-value"}'
   ```

   Salva l'ARN del segreto da utilizzare nei passaggi seguenti.

   Per ulteriori informazioni, consulta Creazione di un segreto nella Guida per l'AWS Secrets Managerutente.

2. Distribuite il componente secret manager (`aws.greengrass.SecretManager`) con il seguente aggiornamento di fusione della configurazione. Specificate l'ARN del segreto che avete creato in precedenza.

   ```
   {
     "cloudSecrets": [
   ```

```
    {
      "arn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyTestGreengrassSecret-abcdef"
    }
  ]
}
```

Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi o il comando di distribuzione della CLI di Greengrass.

3. Crea e distribuisci il componente di esempio in questa sezione con il seguente aggiornamento di configurazione merge. Specificate l'ARN del segreto che avete creato in precedenza.

```
{
  "SecretArn": "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyTestGreengrassSecret",
  "accessControl": {
    "aws.greengrass.SecretManager": {
      "com.example.PrintSecret:secrets:1": {
        "policyDescription": "Allows access to a secret.",
        "operations": [
          "aws.greengrass#GetSecretValue"
        ],
        "resources": [
          "arn:aws:secretsmanager:us-
west-2:123456789012:secret:MyTestGreengrassSecret-abcdef"
        ]
      }
    }
  }
}
```

Per ulteriori informazioni, consultare Crea AWS IoT Greengrass componenti

4. Visualizza i log del software AWS IoT Greengrass Core per verificare che le distribuzioni abbiano esito positivo e visualizza il registro dei `com.example.PrintSecret` componenti per vedere il valore segreto stampato. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Interagisci con le ombre locali

Utilizzate il servizio shadow IPC per interagire con le ombre locali su un dispositivo. Il dispositivo con cui scegli di interagire può essere il tuo dispositivo principale o un dispositivo client connesso.

Per utilizzare queste operazioni IPC, includete il componente shadow manager come dipendenza nel componente personalizzato. È quindi possibile utilizzare le operazioni IPC nei componenti personalizzati per interagire con le ombre locali sul dispositivo tramite lo shadow manager. Per consentire ai componenti personalizzati di reagire ai cambiamenti negli stati shadow locali, puoi anche utilizzare il servizio IPC publish/subscribe per sottoscrivere gli eventi shadow. Per ulteriori informazioni sull'utilizzo del servizio publish/subscribe, consulta la. Pubblicare/sottoscrivere messaggi locali

> **Note**
>
> Per consentire a un dispositivo principale di interagire con le ombre dei dispositivi client, è inoltre necessario configurare e distribuire il componente bridge MQTT. Per ulteriori informazioni, consultate Attivare lo shadow manager per comunicare con i dispositivi client.

Argomenti

- Versioni SDK minime
- Autorizzazione
- GetThingShadow
- UpdateThingShadow
- DeleteThingShadow
- ListNamedShadowsForThing

Versioni SDK minime

La tabella seguente elenca le versioni minime di SDK per dispositivi AWS IoT da utilizzare per interagire con le ombre locali.

SDK	Versione minima	
SDK per dispositivi AWS IoTper Java v2	v1.4.0	
SDK per dispositivi AWS IoTper Python v2	v1.6.0	
SDK per dispositivi AWS IoTper C++ v2	v1.17.0	
SDK per dispositivi AWS IoTper v2 JavaScript	v1.12.0	

Autorizzazione

Per utilizzare il servizio shadow IPC in un componente personalizzato, è necessario definire politiche di autorizzazione che consentano al componente di interagire con le ombre. Per informazioni sulla definizione delle politiche di autorizzazione, vedere. Autorizza i componenti a eseguire operazioni IPC

Le politiche di autorizzazione per l'interazione shadow hanno le seguenti proprietà.

Identificatore del servizio IPC: `aws.greengrass.ShadowManager`

Operazione	Descrizione	Risorse
`aws.greengrass#GetThingShadow`	Consente a un componente di recuperare l'ombra di un oggetto.	Una delle seguenti stringhe: • `$aws/things/` *`thingName`* `/shadow/`, per consentire l'accesso alla classica ombra del dispositivo. • `$aws/things/` *`thingName`* `/shadow/name/` *`shadowName`* ,

Operazione	Descrizione	Risorse
		per consentire l'accesso a un'ombra denominata. • *per consentire l'accesso a tutte le ombre.
`aws.greengrass#UpdateThingShadow`	Consente a un componente di aggiornare l'ombra di un oggetto.	Una delle seguenti stringhe: • `$aws/things/` *`thingName`* `/shadow/`, per consentire l'accesso alla classica ombra del dispositivo. • `$aws/things/` *`thingName`* `/shadow/name/` *`shadowName`* , per consentire l'accesso a un'ombra denominata. • *per consentire l'accesso a tutte le ombre.

Operazione	Descrizione	Risorse
`aws.greengrass#DeleteThingShadow`	Consente a un componente di eliminare l'ombra di un oggetto.	Una delle seguenti stringhe: • `$aws/things/` *`thingName`* `/shadow/`, per consentire l'accesso alla classica ombra del dispositivo • `$aws/things/` *`thingName`* `/shadow/name/` *`shadowName`* , per consentire l'accesso a un'ombra denominata • `*`, per consentire l'accesso a tutte le ombre.
`aws.greengrass#ListNamedShadowsForThing`	Consente a un componente di recuperare l'elenco delle ombre denominate per un oggetto.	Una stringa di nome dell'oggetto che consente di accedere all'oggetto per elencarne le ombre. `*`Da utilizzare per consentire l'accesso a tutte le cose.

Identificatore del servizio IPC: `aws.greengrass.ipc.pubsub`

Operazione	Descrizione	Risorse
`aws.greengrass#SubscribeToTopic`	Consente a un componente di sottoscrivere i messaggi per gli argomenti specificati.	Una delle seguenti stringhe di argomenti: • *`shadowTopicPrefix`* `/get/accepted`

Operazione	Descrizione	Risorse
		• *shadowTopicPrefix* / get/rejected • *shadowTopicPrefix* / delete/accepted • *shadowTopicPrefix* / delete/rejected • *shadowTopicPrefix* / update/accepted • *shadowTopicPrefix* / update/delta • *shadowTopicPrefix* / update/rejected Il valore del prefisso dell'argomento *shadowTopicPrefix* dipende dal tipo di ombra: • Ombra classica: $aws/things/ *thingName* /shadow • Ombra denominata: $aws/things/ *thingName* /shadow/name/ *shadowName* Utilizzato * per consentire l'accesso a tutti gli argomenti. In Greengrass nucleus v2.6.0 e versioni successive, è possibile sottoscrivere argomenti che contengono

Operazione	Descrizione	Risorse
		caratteri jolly degli argomenti MQTT (and). # + Questa stringa di argomenti supporta i caratteri jolly degli argomenti MQTT come caratteri letterali . Ad esempio, se la politica di autorizzazione di un componente concede l'accesso `atest/topic/#` , il componente può sottoscri veretest/topic/# , ma non può sottoscrivere. `test/topic/filter`

Variabili Recipe nelle politiche di autorizzazione shadow locali

Se si utilizza la versione 2.6.0 o successiva del nucleo di Greengrass e si imposta l'opzione di interpolateComponentConfigurationconfigurazione di Greengrass nucleus su`true`, è possibile utilizzare la variabile recipe nelle politiche di autorizzazione. `{iot:thingName}` Questa funzionalità consente di configurare un'unica politica di autorizzazione per un gruppo di dispositivi principali, in cui ogni dispositivo principale può accedere solo alla propria ombra. Ad esempio, è possibile consentire a un componente l'accesso alla seguente risorsa per le operazioni IPC shadow.

```
$aws/things/{iot:thingName}/shadow/
```

Esempi di politiche di autorizzazione

Puoi fare riferimento ai seguenti esempi di politiche di autorizzazione per aiutarti a configurare le politiche di autorizzazione per i tuoi componenti.

Example Esempio: consenti a un gruppo di dispositivi principali di interagire con le ombre locali

> **Important**
>
> Questo esempio utilizza una funzionalità disponibile per la versione 2.6.0 e successive del componente Greengrass nucleus. Greengrass nucleus v2.6.0 aggiunge il supporto

per la maggior parte delle variabili di ricetta, ad esempio nelle configurazioni dei componenti. `{iot:thingName}` Per abilitare questa funzionalità, imposta l'opzione di interpolateComponentConfigurationconfigurazione del nucleo di Greengrass su. `true` Per un esempio che funziona per tutte le versioni del Greengrass nucleus, vedi la politica di autorizzazione di esempio per un dispositivo single core.

Il seguente esempio di politica di autorizzazione consente `com.example.MyShadowInteractionComponent` al componente di interagire con la classica ombra del dispositivo e l'ombra denominata `myNamedShadow` per il dispositivo principale che esegue il componente. Questa politica consente inoltre a questo componente di ricevere messaggi su argomenti locali relativi a queste ombre.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ShadowManager": {
      "com.example.MyShadowInteractionComponent:shadow:1": {
        "policyDescription": "Allows access to shadows",
        "operations": [
          "aws.greengrass#GetThingShadow",
          "aws.greengrass#UpdateThingShadow",
          "aws.greengrass#DeleteThingShadow"
        ],
        "resources": [
          "$aws/things/{iot:thingName}/shadow",
          "$aws/things/{iot:thingName}/shadow/name/myNamedShadow"
        ]
      },
      "com.example.MyShadowInteractionComponent:shadow:2": {
        "policyDescription": "Allows access to things with shadows",
        "operations": [
          "aws.greengrass#ListNamedShadowsForThing"
        ],
        "resources": [
          "{iot:thingName}"
        ]
      }
    },
    "aws.greengrass.ipc.pubsub": {
```

```
      "com.example.MyShadowInteractionComponent:pubsub:1": {
        "policyDescription": "Allows access to shadow pubsub topics",
        "operations": [
          "aws.greengrass#SubscribeToTopic"
        ],
        "resources": [
          "$aws/things/{iot:thingName}/shadow/get/accepted",
          "$aws/things/{iot:thingName}/shadow/name/myNamedShadow/get/accepted"
        ]
      }
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ShadowManager:
    'com.example.MyShadowInteractionComponent:shadow:1':
      policyDescription: 'Allows access to shadows'
      operations:
        - 'aws.greengrass#GetThingShadow'
        - 'aws.greengrass#UpdateThingShadow'
        - 'aws.greengrass#DeleteThingShadow'
      resources:
        - $aws/things/{iot:thingName}/shadow
        - $aws/things/{iot:thingName}/shadow/name/myNamedShadow
    'com.example.MyShadowInteractionComponent:shadow:2':
      policyDescription: 'Allows access to things with shadows'
      operations:
        - 'aws.greengrass#ListNamedShadowsForThing'
      resources:
        - '{iot:thingName}'
  aws.greengrass.ipc.pubsub:
    'com.example.MyShadowInteractionComponent:pubsub:1':
      policyDescription: 'Allows access to shadow pubsub topics'
      operations:
        - 'aws.greengrass#SubscribeToTopic'
      resources:
        - $aws/things/{iot:thingName}/shadow/get/accepted
        - $aws/things/{iot:thingName}/shadow/name/myNamedShadow/get/accepted
```

Example Esempio: consenti a un gruppo di dispositivi principali di interagire con le ombre dei dispositivi client

> **Important**
>
> Questa funzionalità richiede Greengrass nucleus v2.6.0 o successivo, shadow manager v2.2.0 o successivo e MQTT bridge v2.2.0 o successivo. È necessario configurare il bridge MQTT per consentire allo shadow manager di comunicare con i dispositivi client.

Il seguente esempio di politica di autorizzazione consente `com.example.MyShadowInteractionComponent` al componente di interagire con tutte le ombre dei dispositivi per i dispositivi client i cui nomi iniziano con. `MyClientDevice`

> **Note**
>
> Per consentire a un dispositivo principale di interagire con le ombre dei dispositivi client, è inoltre necessario configurare e distribuire il componente bridge MQTT. Per ulteriori informazioni, consultate Attivare lo shadow manager per comunicare con i dispositivi client.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ShadowManager": {
      "com.example.MyShadowInteractionComponent:shadow:1": {
        "policyDescription": "Allows access to shadows",
        "operations": [
          "aws.greengrass#GetThingShadow",
          "aws.greengrass#UpdateThingShadow",
          "aws.greengrass#DeleteThingShadow"
        ],
        "resources": [
          "$aws/things/MyClientDevice*/shadow",
          "$aws/things/MyClientDevice*/shadow/name/*"
        ]
      },
      "com.example.MyShadowInteractionComponent:shadow:2": {
        "policyDescription": "Allows access to things with shadows",
        "operations": [
```

```
          "aws.greengrass#ListNamedShadowsForThing"
        ],
        "resources": [
          "MyClientDevice*"
        ]
      }
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ShadowManager:
    'com.example.MyShadowInteractionComponent:shadow:1':
      policyDescription: 'Allows access to shadows'
      operations:
        - 'aws.greengrass#GetThingShadow'
        - 'aws.greengrass#UpdateThingShadow'
        - 'aws.greengrass#DeleteThingShadow'
      resources:
        - $aws/things/MyClientDevice*/shadow
        - $aws/things/MyClientDevice*/shadow/name/*
    'com.example.MyShadowInteractionComponent:shadow:2':
      policyDescription: 'Allows access to things with shadows'
      operations:
        - 'aws.greengrass#ListNamedShadowsForThing'
      resources:
        - MyClientDevice*
```

Example Esempio: consenti a un dispositivo single-core di interagire con le ombre locali

Il seguente esempio di politica di autorizzazione consente `com.example.MyShadowInteractionComponent` al componente di interagire con la classica ombra del dispositivo e l'ombra denominata `myNamedShadow` per il dispositivo`MyThingName`. Questa politica consente inoltre a questo componente di ricevere messaggi su argomenti locali relativi a queste ombre.

JSON

```
{
```

```
  "accessControl": {
    "aws.greengrass.ShadowManager": {
      "com.example.MyShadowInteractionComponent:shadow:1": {
        "policyDescription": "Allows access to shadows",
        "operations": [
          "aws.greengrass#GetThingShadow",
          "aws.greengrass#UpdateThingShadow",
          "aws.greengrass#DeleteThingShadow"
        ],
        "resources": [
          "$aws/things/MyThingName/shadow",
          "$aws/things/MyThingName/shadow/name/myNamedShadow"
        ]
      },
      "com.example.MyShadowInteractionComponent:shadow:2": {
        "policyDescription": "Allows access to things with shadows",
        "operations": [
          "aws.greengrass#ListNamedShadowsForThing"
        ],
        "resources": [
          "MyThingName"
        ]
      }
    },
    "aws.greengrass.ipc.pubsub": {
      "com.example.MyShadowInteractionComponent:pubsub:1": {
        "policyDescription": "Allows access to shadow pubsub topics",
        "operations": [
          "aws.greengrass#SubscribeToTopic"
        ],
        "resources": [
          "$aws/things/MyThingName/shadow/get/accepted",
          "$aws/things/MyThingName/shadow/name/myNamedShadow/get/accepted"
        ]
      }
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ShadowManager:
```

```
    'com.example.MyShadowInteractionComponent:shadow:1':
      policyDescription: 'Allows access to shadows'
      operations:
        - 'aws.greengrass#GetThingShadow'
        - 'aws.greengrass#UpdateThingShadow'
        - 'aws.greengrass#DeleteThingShadow'
      resources:
        - $aws/things/MyThingName/shadow
        - $aws/things/MyThingName/shadow/name/myNamedShadow
    'com.example.MyShadowInteractionComponent:shadow:2':
      policyDescription: 'Allows access to things with shadows'
      operations:
        - 'aws.greengrass#ListNamedShadowsForThing'
      resources:
        - MyThingName
  aws.greengrass.ipc.pubsub:
    'com.example.MyShadowInteractionComponent:pubsub:1':
      policyDescription: 'Allows access to shadow pubsub topics'
      operations:
        - 'aws.greengrass#SubscribeToTopic'
      resources:
        - $aws/things/MyThingName/shadow/get/accepted
        - $aws/things/MyThingName/shadow/name/myNamedShadow/get/accepted
```

Example Esempio: consenti a un gruppo di dispositivi principali di reagire ai cambiamenti dello stato ombra locale

> **Important**
>
> Questo esempio utilizza una funzionalità disponibile per la versione 2.6.0 e successive del componente Greengrass nucleus. Greengrass nucleus v2.6.0 aggiunge il supporto per la maggior parte delle variabili di ricetta, ad esempio nelle configurazioni dei componenti. `{iot:thingName}` Per abilitare questa funzionalità, imposta l'opzione di interpolateComponentConfigurationconfigurazione del nucleo di Greengrass su. `true` Per un esempio che funziona per tutte le versioni del Greengrass nucleus, vedi la politica di autorizzazione di esempio per un dispositivo single core.

Il seguente esempio di politica di controllo degli accessi consente `com.example.MyShadowReactiveComponent` all'utente personalizzato di ricevere messaggi

sull'`/update/delta`argomento per il dispositivo shadow classico e l'shadow denominato `myNamedShadow` su ogni dispositivo principale su cui è in esecuzione il componente.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.pubsub": {
      "com.example.MyShadowReactiveComponent:pubsub:1": {
        "policyDescription": "Allows access to shadow pubsub topics",
        "operations": [
          "aws.greengrass#SubscribeToTopic"
        ],
        "resources": [
          "$aws/things/{iot:thingName}/shadow/update/delta",
          "$aws/things/{iot:thingName}/shadow/name/myNamedShadow/update/delta"
        ]
      }
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ipc.pubsub:
    "com.example.MyShadowReactiveComponent:pubsub:1":
      policyDescription: Allows access to shadow pubsub topics
      operations:
        - 'aws.greengrass#SubscribeToTopic'
      resources:
        - $aws/things/{iot:thingName}/shadow/update/delta
        - $aws/things/{iot:thingName}/shadow/name/myNamedShadow/update/delta
```

Example Esempio: consentire a un dispositivo single-core di reagire ai cambiamenti dello stato ombra locale

Il seguente esempio di politica di controllo degli accessi consente `com.example.MyShadowReactiveComponent` all'utente personalizzato di ricevere messaggi

sull'`/update/delta`argomento per il dispositivo shadow classico e l'ombra denominata `myNamedShadow` per il dispositivo`MyThingName`.

JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.pubsub": {
      "com.example.MyShadowReactiveComponent:pubsub:1": {
        "policyDescription": "Allows access to shadow pubsub topics",
        "operations": [
          "aws.greengrass#SubscribeToTopic"
        ],
        "resources": [
          "$aws/things/MyThingName/shadow/update/delta",
          "$aws/things/MyThingName/shadow/name/myNamedShadow/update/delta"
        ]
      }
    }
  }
}
```

YAML

```
accessControl:
  aws.greengrass.ipc.pubsub:
    "com.example.MyShadowReactiveComponent:pubsub:1":
      policyDescription: Allows access to shadow pubsub topics
      operations:
        - 'aws.greengrass#SubscribeToTopic'
      resources:
        - $aws/things/MyThingName/shadow/update/delta
        - $aws/things/MyThingName/shadow/name/myNamedShadow/update/delta
```

GetThingShadow

Ottieni l'ombra per una cosa specificata.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`thingName`(Python:) `thing_name`

Nome dell'oggetto.

Tipo: `string`

`shadowName`(Python:) `shadow_name`

Il nome della copia shadow. Per specificare l'ombra classica dell'oggetto, imposta questo parametro su una stringa vuota (`""`).

> **Warning**
>
> Il AWS IoT Greengrass servizio utilizza lo shadow `AWSManagedGreengrassV2Deployment` denominato per gestire le distribuzioni destinate a singoli dispositivi core. Questa ombra denominata è riservata all'uso da parte del AWS IoT Greengrass servizio. Non aggiornare o eliminare questa ombra denominata.

Tipo: `string`

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`payload`

Il documento sullo stato della risposta sotto forma di blob.

Tipo: `object` che contiene le seguenti informazioni:

`state`

Le informazioni sullo stato.

Questo oggetto contiene le seguenti informazioni.

`desired`

Le proprietà e i valori dello stato da aggiornare nel dispositivo.

Tipo: `map` di coppie chiave-valore

`reported`

Le proprietà e i valori dello stato riportati dal dispositivo.

Tipo: `map` di coppie chiave-valore

`delta`

La differenza tra le proprietà e i valori dello stato desiderati e quelli riportati. Questa proprietà è presente solo se gli `reported` stati `desired` and sono diversi.

Tipo: `map` di coppie chiave-valore

`metadata`

I timestamp per ogni attributo nelle `reported` sezioni `desired` and in modo da poter determinare quando lo stato è stato aggiornato.

Tipo: `string`

`timestamp`

L'epoca, la data e l'ora in cui è stata generata la risposta.

Tipo: `integer`

`clientToken`(Python:) `clientToken`

Il token utilizzato per abbinare la richiesta e la risposta corrispondente

Tipo: `string`

`version`

La versione del documento shadow locale.

Tipo: `integer`

Errori

Questa operazione può restituire i seguenti errori.

`InvalidArgumentsError`

Il servizio shadow locale non è in grado di convalidare i parametri della richiesta. Ciò può verificarsi se la richiesta contiene caratteri JSON non validi o non supportati.

`ResourceNotFoundError`

Il documento shadow locale richiesto non può essere trovato.

`ServiceError`

Si è verificato un errore interno del servizio oppure il numero di richieste al servizio IPC ha superato i limiti specificati nei parametri di `maxTotalLocalRequestsRate` configurazione `maxLocalRequestsPerSecondPerThing` e nel componente shadow manager.

`UnauthorizedError`

La politica di autorizzazione del componente non include le autorizzazioni necessarie per questa operazione.

Esempi

Gli esempi seguenti mostrano come chiamare questa operazione nel codice del componente personalizzato.

Java (IPC client V1)

Example Esempio: Get a thing shadow

> **Note**
>
> Questo esempio utilizza una `IPCUtils` classe per creare una connessione al servizio AWS IoT Greengrass Core IPC. Per ulteriori informazioni, consulta Connect al servizio AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GetThingShadowResponseHandler;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.model.GetThingShadowRequest;
import software.amazon.awssdk.aws.greengrass.model.GetThingShadowResponse;
import software.amazon.awssdk.aws.greengrass.model.ResourceNotFoundError;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;
```

```
import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class GetThingShadow {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        // Use the current core device's name if thing name isn't set.
        String thingName = args[0].isEmpty() ? System.getenv("AWS_IOT_THING_NAME") :
 args[0];
        String shadowName = args[1];
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            GetThingShadowResponseHandler responseHandler =
                    GetThingShadow.getThingShadow(ipcClient, thingName,
 shadowName);
            CompletableFuture<GetThingShadowResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                GetThingShadowResponse response =
 futureResponse.get(TIMEOUT_SECONDS,
                        TimeUnit.SECONDS);
                String shadowPayload = new String(response.getPayload(),
 StandardCharsets.UTF_8);
                System.out.printf("Successfully got shadow %s/%s: %s%n", thingName,
 shadowName,
                        shadowPayload);
            } catch (TimeoutException e) {
                System.err.printf("Timeout occurred while getting shadow: %s/%s%n",
 thingName,
                        shadowName);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.printf("Unauthorized error while getting shadow: %s/
%s%n",
                            thingName, shadowName);
                } else if (e.getCause() instanceof ResourceNotFoundError) {
```

```
                    System.err.printf("Unable to find shadow to get: %s/%s%n",
 thingName,
                            shadowName);
                } else {
                    throw e;
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static GetThingShadowResponseHandler
 getThingShadow(GreengrassCoreIPCClient greengrassCoreIPCClient, String thingName,
 String shadowName) {
        GetThingShadowRequest getThingShadowRequest = new GetThingShadowRequest();
        getThingShadowRequest.setThingName(thingName);
        getThingShadowRequest.setShadowName(shadowName);
        return greengrassCoreIPCClient.getThingShadow(getThingShadowRequest,
 Optional.empty());
    }
}
```

Python (IPC client V1)

Example Esempio: Get a thing shadow

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import GetThingShadowRequest

TIMEOUT = 10

def sample_get_thing_shadow_request(thingName, shadowName):
    try:
        # set up IPC client to connect to the IPC server
        ipc_client = awsiot.greengrasscoreipc.connect()

        # create the GetThingShadow request
        get_thing_shadow_request = GetThingShadowRequest()
```

```
        get_thing_shadow_request.thing_name = thingName
        get_thing_shadow_request.shadow_name = shadowName

        # retrieve the GetThingShadow response after sending the request to the IPC
 server
        op = ipc_client.new_get_thing_shadow()
        op.activate(get_thing_shadow_request)
        fut = op.get_response()

        result = fut.result(TIMEOUT)
        return result.payload

    except InvalidArgumentsError as e:
        # add error handling
        ...
    # except ResourceNotFoundError | UnauthorizedError | ServiceError
```

JavaScript

Example Esempio: ottieni un'ombra per una cosa

```
import {
    GetThingShadowRequest
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc';

class GetThingShadow {
    private ipcClient: greengrasscoreipc.Client;
    private thingName: string;
    private shadowName: string;

    constructor() {
        // Define args parameters here
        this.thingName = "<define_your_own_thingName>";
        this.shadowName = "<define_your_own_shadowName>";
        this.bootstrap();
    }

    async bootstrap() {
        try {
            this.ipcClient = await getIpcClient();
        } catch (err) {
            // parse the error depending on your use cases
            throw err
```

```
        }

        try {
            await this.handleGetThingShadowOperation(this.thingName,
                this.shadowName);
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }
    }

    async handleGetThingShadowOperation(
        thingName: string,
        shadowName: string
    ) {
        const request: GetThingShadowRequest = {
            thingName: thingName,
            shadowName: shadowName
        };
        const response = await this.ipcClient.getThingShadow(request);
    }
}

export async function getIpcClient() {
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use caseså
        throw err
    }
}

const startScript = new GetThingShadow();
```

UpdateThingShadow

Aggiorna l'ombra per l'oggetto specificato. Se non esiste un'ombra, ne viene creata una.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`thingName`(Python:) `thing_name`

Nome dell'oggetto.

Tipo: `string`

`shadowName`(Python:) `shadow_name`

Il nome della copia shadow. Per specificare l'ombra classica dell'oggetto, imposta questo parametro su una stringa vuota (`""`).

> **Warning**
>
> Il AWS IoT Greengrass servizio utilizza lo shadow `AWSManagedGreengrassV2Deployment` denominato per gestire le distribuzioni destinate a singoli dispositivi core. Questa ombra denominata è riservata all'uso da parte del AWS IoT Greengrass servizio. Non aggiornare o eliminare questa ombra denominata.

Tipo: `string`

`payload`

Il documento di stato della richiesta come blob.

Tipo: `object` che contiene le seguenti informazioni:

`state`

Le informazioni sullo stato da aggiornare. Questa operazione IPC ha effetto solo sui campi specificati.

Questo oggetto contiene le seguenti informazioni. In genere, utilizzerai la `desired` proprietà o la `reported` proprietà, ma non entrambe nella stessa richiesta.

`desired`

Le proprietà e i valori dello stato di cui si richiede l'aggiornamento nel dispositivo.

Tipo: `map` di coppie chiave-valore

`reported`

Le proprietà e i valori dello stato riportati dal dispositivo.

Tipo: `map` di coppie chiave-valore

`clientToken`(Python:) `client_token`

(Facoltativo) Il token utilizzato per abbinare la richiesta e la risposta corrispondente dal token client.

Tipo: `string`

`version`

(Facoltativo) La versione del documento shadow locale da aggiornare. Il servizio shadow elabora l'aggiornamento solo se la versione specificata corrisponde all'ultima versione disponibile.

Tipo: `integer`

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`payload`

Il documento sullo stato della risposta sotto forma di blob.

Tipo: `object` che contiene le seguenti informazioni:

`state`

Le informazioni sullo stato.

Questo oggetto contiene le seguenti informazioni.

`desired`

Le proprietà e i valori dello stato da aggiornare nel dispositivo.

Tipo: `map` di coppie chiave-valore

`reported`

Le proprietà e i valori dello stato riportati dal dispositivo.

Tipo: `map` di coppie chiave-valore

`delta`

Le proprietà e i valori dello stato riportati dal dispositivo.

Tipo: `map` di coppie chiave-valore

`metadata`

I timestamp per ogni attributo nelle `reported` sezioni `desired` and in modo da poter determinare quando lo stato è stato aggiornato.

Tipo: `string`

`timestamp`

L'epoca, la data e l'ora in cui è stata generata la risposta.

Tipo: `integer`

`clientToken`(Python:) `client_token`

Il token utilizzato per abbinare la richiesta e la risposta corrispondente.

Tipo: `string`

`version`

La versione del documento shadow locale dopo il completamento dell'aggiornamento.

Tipo: `integer`

Errori

Questa operazione può restituire i seguenti errori.

`ConflictError`

Il servizio shadow locale ha rilevato un conflitto di versione durante l'operazione di aggiornamento. Ciò si verifica quando la versione nel payload della richiesta non corrisponde alla versione dell'ultimo documento shadow locale disponibile.

`InvalidArgumentsError`

Il servizio shadow locale non è in grado di convalidare i parametri della richiesta. Ciò può verificarsi se la richiesta contiene caratteri JSON non validi o non supportati.

Un valido `payload` ha le seguenti proprietà:

- Il `state` nodo esiste ed è un oggetto che contiene le informazioni `reported` sullo stato `desired` o.
- I `reported` nodi `desired` and sono oggetti o nulli. Almeno uno di questi oggetti deve contenere informazioni sullo stato valide.
- La profondità degli `reported` oggetti `desired` e non può superare gli otto nodi.
- La lunghezza del `clientToken` valore non può superare i 64 caratteri.
- Il `version` valore deve essere uguale 1 o superiore.

`ServiceError`

Si è verificato un errore interno del servizio oppure il numero di richieste al servizio IPC ha superato i limiti specificati nei parametri di `maxTotalLocalRequestsRate` configurazione `maxLocalRequestsPerSecondPerThing` e nel componente shadow manager.

`UnauthorizedError`

La politica di autorizzazione del componente non include le autorizzazioni necessarie per questa operazione.

Esempi

Gli esempi seguenti mostrano come chiamare questa operazione nel codice del componente personalizzato.

Java (IPC client V1)

Example Esempio: aggiorna l'ombra di una cosa

> **Note**
>
> Questo esempio utilizza una `IPCUtils` classe per creare una connessione al servizio AWS IoT Greengrass Core IPC. Per ulteriori informazioni, consulta Connect al servizio AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.UpdateThingShadowResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.aws.greengrass.model.UpdateThingShadowRequest;
import software.amazon.awssdk.aws.greengrass.model.UpdateThingShadowResponse;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.nio.charset.StandardCharsets;
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class UpdateThingShadow {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        // Use the current core device's name if thing name isn't set.
        String thingName = args[0].isEmpty() ? System.getenv("AWS_IOT_THING_NAME") :
 args[0];
        String shadowName = args[1];
        byte[] shadowPayload = args[2].getBytes(StandardCharsets.UTF_8);
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            UpdateThingShadowResponseHandler responseHandler =
                    UpdateThingShadow.updateThingShadow(ipcClient, thingName,
 shadowName,
                            shadowPayload);
            CompletableFuture<UpdateThingShadowResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                System.out.printf("Successfully updated shadow: %s/%s%n", thingName,
 shadowName);
            } catch (TimeoutException e) {
```

```
                System.err.printf("Timeout occurred while updating shadow: %s/%s%n",
 thingName,
                        shadowName);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.printf("Unauthorized error while updating shadow: %s/
%s%n",
                            thingName, shadowName);
                } else {
                    throw e;
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static UpdateThingShadowResponseHandler
 updateThingShadow(GreengrassCoreIPCClient greengrassCoreIPCClient, String
 thingName, String shadowName, byte[] shadowPayload) {
        UpdateThingShadowRequest updateThingShadowRequest = new
 UpdateThingShadowRequest();
        updateThingShadowRequest.setThingName(thingName);
        updateThingShadowRequest.setShadowName(shadowName);
        updateThingShadowRequest.setPayload(shadowPayload);
        return greengrassCoreIPCClient.updateThingShadow(updateThingShadowRequest,
                Optional.empty());
    }
}
```

Python (IPC client V1)

Example Esempio: aggiorna l'ombra di una cosa

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import UpdateThingShadowRequest

TIMEOUT = 10
```

```
def sample_update_thing_shadow_request(thingName, shadowName, payload):
    try:
        # set up IPC client to connect to the IPC server
        ipc_client = awsiot.greengrasscoreipc.connect()

        # create the UpdateThingShadow request
        update_thing_shadow_request = UpdateThingShadowRequest()
        update_thing_shadow_request.thing_name = thingName
        update_thing_shadow_request.shadow_name = shadowName
        update_thing_shadow_request.payload = payload

        # retrieve the UpdateThingShadow response after sending the request to the
 IPC server
        op = ipc_client.new_update_thing_shadow()
        op.activate(update_thing_shadow_request)
        fut = op.get_response()

        result = fut.result(TIMEOUT)
        return result.payload

    except InvalidArgumentsError as e:
        # add error handling
    ...
    # except ConflictError | UnauthorizedError | ServiceError
```

JavaScript

Example Esempio: aggiornare l'ombra di una cosa

```
import {
    UpdateThingShadowRequest
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc';

class UpdateThingShadow {
    private ipcClient: greengrasscoreipc.Client;
    private thingName: string;
    private shadowName: string;
    private shadowDocumentStr: string;

    constructor() {
        // Define args parameters here

        this.thingName = "<define_your_own_thingName>";
```

```
        this.shadowName = "<define_your_own_shadowName>";
        this.shadowDocumentStr = "<define_your_own_payload>";

        this.bootstrap();
    }

    async bootstrap() {
        try {
            this.ipcClient = await getIpcClient();
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }

        try {
            await this.handleUpdateThingShadowOperation(
                this.thingName,
                this.shadowName,
                this.shadowDocumentStr);
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }
    }

    async handleUpdateThingShadowOperation(
        thingName: string,
        shadowName: string,
        payloadStr: string
    ) {
        const request: UpdateThingShadowRequest = {
            thingName: thingName,
            shadowName: shadowName,
            payload: payloadStr
        }
        // make the UpdateThingShadow request
        const response = await this.ipcClient.updateThingShadow(request);
    }
}

export async function getIpcClient() {
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
```

```
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

const startScript = new UpdateThingShadow();
```

DeleteThingShadow

Elimina la copia shadow per l'oggetto specificato.

A partire da shadow manager v2.0.4, l'eliminazione di un'ombra incrementa il numero di versione. Ad esempio, quando si elimina l'ombra `MyThingShadow` nella versione 1, la versione dell'ombra eliminata è 2. Se poi ricreate un'ombra con il nome`MyThingShadow`, la versione per quell'ombra è 3.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`thingName`(Python:) `thing_name`

Nome dell'oggetto.

Tipo: `string`

`shadowName`(Python:) `shadow_name`

Il nome della copia shadow. Per specificare l'ombra classica dell'oggetto, imposta questo parametro su una stringa vuota (`""`).

> **Warning**
>
> Il AWS IoT Greengrass servizio utilizza lo shadow `AWSManagedGreengrassV2Deployment` denominato per gestire le distribuzioni

destinate a singoli dispositivi core. Questa ombra denominata è riservata all'uso da parte del AWS IoT Greengrass servizio. Non aggiornare o eliminare questa ombra denominata.

Tipo: `string`

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`payload`

Un documento vuoto sullo stato della risposta.

Errori

Questa operazione può restituire i seguenti errori.

`InvalidArgumentsError`

Il servizio shadow locale non è in grado di convalidare i parametri della richiesta. Ciò può verificarsi se la richiesta contiene caratteri JSON non validi o non supportati.

`ResourceNotFoundError`

Il documento shadow locale richiesto non può essere trovato.

`ServiceError`

Si è verificato un errore interno del servizio oppure il numero di richieste al servizio IPC ha superato i limiti specificati nei parametri di `maxTotalLocalRequestsRate` configurazione `maxLocalRequestsPerSecondPerThing` e nel componente shadow manager.

`UnauthorizedError`

La politica di autorizzazione del componente non include le autorizzazioni necessarie per questa operazione.

Esempi

Gli esempi seguenti mostrano come chiamare questa operazione nel codice del componente personalizzato.

Java (IPC client V1)

Example Esempio: eliminare l'ombra di una cosa

> **Note**
>
> Questo esempio utilizza una `IPCUtils` classe per creare una connessione al servizio AWS IoT Greengrass Core IPC. Per ulteriori informazioni, consulta Connect al servizio AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.DeleteThingShadowResponseHandler;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import software.amazon.awssdk.aws.greengrass.model.DeleteThingShadowRequest;
import software.amazon.awssdk.aws.greengrass.model.DeleteThingShadowResponse;
import software.amazon.awssdk.aws.greengrass.model.ResourceNotFoundError;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class DeleteThingShadow {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        // Use the current core device's name if thing name isn't set.
        String thingName = args[0].isEmpty() ? System.getenv("AWS_IOT_THING_NAME") :
 args[0];
        String shadowName = args[1];
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            DeleteThingShadowResponseHandler responseHandler =
```

```
                    DeleteThingShadow.deleteThingShadow(ipcClient, thingName,
 shadowName);
            CompletableFuture<DeleteThingShadowResponse> futureResponse =
                    responseHandler.getResponse();
            try {
                futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                System.out.printf("Successfully deleted shadow: %s/%s%n", thingName,
 shadowName);
            } catch (TimeoutException e) {
                System.err.printf("Timeout occurred while deleting shadow: %s/%s%n",
 thingName,
                        shadowName);
            } catch (ExecutionException e) {
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.printf("Unauthorized error while deleting shadow: %s/
%s%n",
                            thingName, shadowName);
                } else if (e.getCause() instanceof ResourceNotFoundError) {
                    System.err.printf("Unable to find shadow to delete: %s/%s%n",
 thingName,
                            shadowName);
                } else {
                    throw e;
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static DeleteThingShadowResponseHandler
 deleteThingShadow(GreengrassCoreIPCClient greengrassCoreIPCClient, String
 thingName, String shadowName) {
        DeleteThingShadowRequest deleteThingShadowRequest = new
 DeleteThingShadowRequest();
        deleteThingShadowRequest.setThingName(thingName);
        deleteThingShadowRequest.setShadowName(shadowName);
        return greengrassCoreIPCClient.deleteThingShadow(deleteThingShadowRequest,
                Optional.empty());
    }
```

```
}
```

Python (IPC client V1)

Example Esempio: eliminare un'ombra di oggetto

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import DeleteThingShadowRequest

TIMEOUT = 10

def sample_delete_thing_shadow_request(thingName, shadowName):
    try:
        # set up IPC client to connect to the IPC server
        ipc_client = awsiot.greengrasscoreipc.connect()

        # create the DeleteThingShadow request
        delete_thing_shadow_request = DeleteThingShadowRequest()
        delete_thing_shadow_request.thing_name = thingName
        delete_thing_shadow_request.shadow_name = shadowName

        # retrieve the DeleteThingShadow response after sending the request to the
 IPC server
        op = ipc_client.new_delete_thing_shadow()
        op.activate(delete_thing_shadow_request)
        fut = op.get_response()

        result = fut.result(TIMEOUT)
        return result.payload

    except InvalidArgumentsError as e:
        # add error handling
    ...
    # except ResourceNotFoundError | UnauthorizedError | ServiceError
```

JavaScript

Example Esempio: eliminare l'ombra di una cosa

```
import {
    DeleteThingShadowRequest
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc';
```

```
class DeleteThingShadow {
    private ipcClient: greengrasscoreipc.Client;
    private thingName: string;
    private shadowName: string;

    constructor() {
        // Define args parameters here
        this.thingName = "<define_your_own_thingName>";
        this.shadowName = "<define_your_own_shadowName>";
        this.bootstrap();
    }

    async bootstrap() {
        try {
            this.ipcClient = await getIpcClient();
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }

        try {
            await this.handleDeleteThingShadowOperation(this.thingName,
 this.shadowName)
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }
    }

    async handleDeleteThingShadowOperation(thingName: string, shadowName: string) {
        const request: DeleteThingShadowRequest = {
            thingName: thingName,
            shadowName: shadowName
        }
        // make the DeleteThingShadow request
        const response = await this.ipcClient.deleteThingShadow(request);
    }
}

export async function getIpcClient() {
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
```

```
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

const startScript = new DeleteThingShadow();
```

ListNamedShadowsForThing

Elenca le ombre denominate per l'oggetto specificato.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`thingName`(Python:) `thing_name`

Nome dell'oggetto.

Tipo: `string`

`pageSize`(Python:) `page_size`

(Facoltativo) Il numero di nomi shadow da restituire in ogni chiamata.

Tipo: `integer`

Impostazione predefinita: 25

Massimo: 100

`nextToken`(Python:) `next_token`

(Facoltativo) Il token per recuperare il prossimo set di risultati. Questo valore viene restituito sui risultati di paging e viene utilizzato nella chiamata che restituisce la pagina successiva.

Tipo: `string`

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`results`

L'elenco dei nomi delle ombre.

Tipo: `array`

`timestamp`

(Facoltativo) La data e l'ora in cui è stata generata la risposta.

Tipo: `integer`

`nextToken`(Python:) `next_token`

(Facoltativo) Il valore del token da utilizzare nelle richieste paginate per recuperare la pagina successiva nella sequenza. Questo token non è presente quando non ci sono più nomi ombra da restituire.

Tipo: `string`

> **Note**
>
> Se la dimensione della pagina richiesta corrisponde esattamente al numero di nomi shadow nella risposta, allora questo token è presente; tuttavia, se utilizzato, restituisce un elenco vuoto.

Errori

Questa operazione può restituire i seguenti errori.

`InvalidArgumentsError`

Il servizio shadow locale non è in grado di convalidare i parametri della richiesta. Ciò può verificarsi se la richiesta contiene caratteri JSON non validi o non supportati.

`ResourceNotFoundError`

Il documento shadow locale richiesto non può essere trovato.

`ServiceError`

Si è verificato un errore interno del servizio oppure il numero di richieste al servizio IPC ha superato i limiti specificati nei parametri di `maxTotalLocalRequestsRate` configurazione `maxLocalRequestsPerSecondPerThing` e nel componente shadow manager.

`UnauthorizedError`

La politica di autorizzazione del componente non include le autorizzazioni necessarie per questa operazione.

Esempi

Gli esempi seguenti mostrano come chiamare questa operazione nel codice del componente personalizzato.

Java (IPC client V1)

Example Esempio: elenca una cosa chiamata shadows

> **Note**
>
> Questo esempio utilizza una `IPCUtils` classe per creare una connessione al servizio AWS IoT Greengrass Core IPC. Per ulteriori informazioni, consulta Connect al servizio AWS IoT Greengrass Core IPC.

```
package com.aws.greengrass.docs.samples.ipc;

import com.aws.greengrass.docs.samples.ipc.util.IPCUtils;
import software.amazon.awssdk.aws.greengrass.GreengrassCoreIPCClient;
import
 software.amazon.awssdk.aws.greengrass.ListNamedShadowsForThingResponseHandler;
import software.amazon.awssdk.aws.greengrass.model.ListNamedShadowsForThingRequest;
import
 software.amazon.awssdk.aws.greengrass.model.ListNamedShadowsForThingResponse;
import software.amazon.awssdk.aws.greengrass.model.ResourceNotFoundError;
import software.amazon.awssdk.aws.greengrass.model.UnauthorizedError;
import software.amazon.awssdk.eventstreamrpc.EventStreamRPCConnection;

import java.util.ArrayList;
import java.util.List;
```

```
import java.util.Optional;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.TimeUnit;
import java.util.concurrent.TimeoutException;

public class ListNamedShadowsForThing {

    public static final int TIMEOUT_SECONDS = 10;

    public static void main(String[] args) {
        // Use the current core device's name if thing name isn't set.
        String thingName = args[0].isEmpty() ? System.getenv("AWS_IOT_THING_NAME") :
 args[0];
        try (EventStreamRPCConnection eventStreamRPCConnection =
                     IPCUtils.getEventStreamRpcConnection()) {
            GreengrassCoreIPCClient ipcClient =
                    new GreengrassCoreIPCClient(eventStreamRPCConnection);
            List<String> namedShadows = new ArrayList<>();
            String nextToken = null;
            try {
                // Send additional requests until there's no pagination token in the
 response.
                do {
                    ListNamedShadowsForThingResponseHandler responseHandler =

 ListNamedShadowsForThing.listNamedShadowsForThing(ipcClient, thingName,
                                    nextToken, 25);
                    CompletableFuture<ListNamedShadowsForThingResponse>
 futureResponse =
                            responseHandler.getResponse();
                    ListNamedShadowsForThingResponse response =
                            futureResponse.get(TIMEOUT_SECONDS, TimeUnit.SECONDS);
                    List<String> responseNamedShadows = response.getResults();
                    namedShadows.addAll(responseNamedShadows);
                    nextToken = response.getNextToken();
                } while (nextToken != null);
                System.out.printf("Successfully got named shadows for thing %s: %s
%n", thingName,
                        String.join(",", namedShadows));
            } catch (TimeoutException e) {
                System.err.println("Timeout occurred while listing named shadows for
 thing: " + thingName);
            } catch (ExecutionException e) {
```

```
                if (e.getCause() instanceof UnauthorizedError) {
                    System.err.println("Unauthorized error while listing named
 shadows for " +
                            "thing: " + thingName);
                } else if (e.getCause() instanceof ResourceNotFoundError) {
                    System.err.println("Unable to find thing to list named shadows:
 " + thingName);
                } else {
                    throw e;
                }
            }
        } catch (InterruptedException e) {
            System.out.println("IPC interrupted.");
        } catch (ExecutionException e) {
            System.err.println("Exception occurred when using IPC.");
            e.printStackTrace();
            System.exit(1);
        }
    }

    public static ListNamedShadowsForThingResponseHandler
 listNamedShadowsForThing(GreengrassCoreIPCClient greengrassCoreIPCClient, String
 thingName, String nextToken, int pageSize) {
        ListNamedShadowsForThingRequest listNamedShadowsForThingRequest =
                new ListNamedShadowsForThingRequest();
        listNamedShadowsForThingRequest.setThingName(thingName);
        listNamedShadowsForThingRequest.setNextToken(nextToken);
        listNamedShadowsForThingRequest.setPageSize(pageSize);
        return
 greengrassCoreIPCClient.listNamedShadowsForThing(listNamedShadowsForThingRequest,
                Optional.empty());
    }
}
```

Python (IPC client V1)

Example Esempio: elenca le ombre di una cosa

```
import awsiot.greengrasscoreipc
import awsiot.greengrasscoreipc.client as client
from awsiot.greengrasscoreipc.model import ListNamedShadowsForThingRequest

TIMEOUT = 10
```

```
def sample_list_named_shadows_for_thing_request(thingName, nextToken, pageSize):
    try:
        # set up IPC client to connect to the IPC server
        ipc_client = awsiot.greengrasscoreipc.connect()

        # create the ListNamedShadowsForThingRequest request
        list_named_shadows_for_thing_request = ListNamedShadowsForThingRequest()
        list_named_shadows_for_thing_request.thing_name = thingName
        list_named_shadows_for_thing_request.next_token = nextToken
        list_named_shadows_for_thing_request.page_size = pageSize

        # retrieve the ListNamedShadowsForThingRequest response after sending the
 request to the IPC server
        op = ipc_client.new_list_named_shadows_for_thing()
        op.activate(list_named_shadows_for_thing_request)
        fut = op.get_response()

        list_result = fut.result(TIMEOUT)

        # additional returned fields
        timestamp = list_result.timestamp
        next_token = result.next_token
        named_shadow_list = list_result.results

        return named_shadow_list, next_token, timestamp

    except InvalidArgumentsError as e:
        # add error handling
    ...
    # except ResourceNotFoundError | UnauthorizedError | ServiceError
```

JavaScript

Example Esempio: elenca una cosa chiamata ombre

```
import {
    ListNamedShadowsForThingRequest
} from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc/model';
import * as greengrasscoreipc from 'aws-iot-device-sdk-v2/dist/greengrasscoreipc';

class listNamedShadowsForThing {
    private ipcClient: greengrasscoreipc.Client;
    private thingName: string;
    private pageSizeStr: string;
```

```
    private nextToken: string;

    constructor() {
        // Define args parameters here
        this.thingName = "<define_your_own_thingName>";
        this.pageSizeStr = "<define_your_own_pageSize>";
        this.nextToken = "<define_your_own_token>";
        this.bootstrap();
    }

    async bootstrap() {
        try {
            this.ipcClient = await getIpcClient();
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }

        try {
            await this.handleListNamedShadowsForThingOperation(this.thingName,
                this.nextToken, this.pageSizeStr);
        } catch (err) {
            // parse the error depending on your use cases
            throw err
        }
    }

    async handleListNamedShadowsForThingOperation(
        thingName: string,
        nextToken: string,
        pageSizeStr: string
    ) {
        let request: ListNamedShadowsForThingRequest = {
            thingName: thingName,
            nextToken: nextToken,
        };
        if (pageSizeStr) {
            request.pageSize = parseInt(pageSizeStr);
        }
        // make the ListNamedShadowsForThing request
        const response = await this.ipcClient.listNamedShadowsForThing(request);
        const shadowNames = response.results;
    }
}
```

```
export async function getIpcClient(){
    try {
        const ipcClient = greengrasscoreipc.createClient();
        await ipcClient.connect()
            .catch(error => {
                // parse the error depending on your use cases
                throw error;
            });
        return ipcClient
    } catch (err) {
        // parse the error depending on your use cases
        throw err
    }
}

const startScript = new listNamedShadowsForThing();
```

Gestisci le implementazioni e i componenti locali

> **Note**
>
> Questa funzionalità è disponibile per la versione 2.6.0 e successive del componente Greengrass nucleus.

Utilizza il servizio IPC Greengrass CLI per gestire le distribuzioni locali e i componenti Greengrass sul dispositivo principale.

Per utilizzare queste operazioni IPC, includete la versione 2.6.0 o successiva del componente Greengrass CLI come dipendenza nel componente personalizzato. È quindi possibile utilizzare le operazioni IPC nei componenti personalizzati per effettuare le seguenti operazioni:

- Crea distribuzioni locali per modificare e configurare i componenti Greengrass sul dispositivo principale.
- Riavvia e arresta i componenti Greengrass sul dispositivo principale.
- Genera una password che puoi usare per accedere alla console di debug locale.

Argomenti

Versioni SDK minime

La tabella seguente elenca le versioni minime di SDK per dispositivi AWS IoT che è necessario utilizzare per interagire con il servizio IPC CLI Greengrass.

SDK	Versione minima	
SDK per dispositivi AWS IoTper Java v2	v1.2.10	
SDK per dispositivi AWS IoTper Python v2	v1.5.3	
SDK per dispositivi AWS IoTper C++ v2	v1.17.0	
SDK per dispositivi AWS IoTper v2 JavaScript	v1.12.0	

Autorizzazione

Per utilizzare il servizio IPC Greengrass CLI in un componente personalizzato, è necessario definire politiche di autorizzazione che consentano al componente di gestire distribuzioni e componenti locali.

Per informazioni sulla definizione delle politiche di autorizzazione, vedere. Autorizza i componenti a eseguire operazioni IPC

Le politiche di autorizzazione per la CLI Greengrass hanno le seguenti proprietà.

Identificatore del servizio IPC: `aws.greengrass.Cli`

Operazione	Descrizione	Risorse
`aws.greengrass#CreateLocalDeployment`	Consente a un componente di creare una distribuzione locale sul dispositivo principale.	*
`aws.greengrass#ListLocalDeployments`	Consente a un componente di elencare le distribuzioni locali sul dispositivo principale.	*
`aws.greengrass#GetLocalDeploymentStatus`	Consente a un componente di ottenere lo stato di una distribuzione locale sul dispositivo principale.	Un ID di distribuzione locale o * per consentire l'accesso a tutte le distribuzioni locali.
`aws.greengrass#ListComponents`	Consente a un componente di elencare i componenti sul dispositivo principale.	*
`aws.greengrass#GetComponentDetails`	Consente a un componente di ottenere dettagli su un componente del dispositivo principale.	Un nome di componente, ad esempio`com.example.HelloWorld` , o * per consentire l'accesso a tutti i componenti.
`aws.greengrass#RestartComponent`	Consente a un componente di riavviare un componente sul dispositivo principale.	Un nome di componente, ad esempio`com.example.HelloWorld` , o * per consentire l'accesso a tutti i componenti.

Operazione	Descrizione	Risorse
`aws.greengrass#StopComponent`	Consente a un componente di arrestare un componente sul dispositivo principale.	Un nome di componente, ad esempio `com.example.HelloWorld` , o * per consentire l'accesso a tutti i componenti.
`aws.greengrass#CreateDebugPassword`	Consente a un componente di generare una password da utilizzare per accedere al componente della console di debug locale.	*

Example Esempio di politica di autorizzazione

I seguenti esempi di politiche di autorizzazione consentono a un componente di creare distribuzioni locali, visualizzare tutte le distribuzioni e i componenti locali e riavviare e arrestare un componente denominato. `com.example.HelloWorld`

```
{
  "accessControl": {
    "aws.greengrass.Cli": {
      "com.example.MyLocalManagerComponent:cli:1": {
        "policyDescription": "Allows access to create local deployments and view
 deployments and components.",
        "operations": [
          "aws.greengrass#CreateLocalDeployment",
          "aws.greengrass#ListLocalDeployments",
          "aws.greengrass#GetLocalDeploymentStatus",
          "aws.greengrass#ListComponents",
          "aws.greengrass#GetComponentDetails"
        ],
        "resources": [
          "*"
        ]
      }
    },
    "aws.greengrass.Cli": {
      "com.example.MyLocalManagerComponent:cli:2": {
```

```
        "policyDescription": "Allows access to restart and stop the Hello World
 component.",
        "operations": [
          "aws.greengrass#RestartComponent",
          "aws.greengrass#StopComponent"
        ],
        "resources": [
          "com.example.HelloWorld"
        ]
      }
    }
  }
}
```

CreateLocalDeployment

Crea o aggiorna una distribuzione locale utilizzando ricette di componenti, artefatti e argomenti di runtime specifici.

Questa operazione fornisce le stesse funzionalità del comando deployment create nella CLI di Greengrass.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`recipeDirectoryPath`(Python:) `recipe_directory_path`

(Facoltativo) Il percorso assoluto della cartella che contiene i file di ricette dei componenti.

`artifactDirectoryPath`(Python:) `artifact_directory_path`

(Facoltativo) Il percorso assoluto della cartella che contiene i file degli artefatti da includere nella distribuzione. La cartella artifacts deve contenere la seguente struttura di cartelle:

```
/path/to/artifact/folder/component-name/component-version/artifacts
```

`rootComponentVersionsToAdd`(Python:) `root_component_versions_to_add`

(Facoltativo) Le versioni dei componenti da installare sul dispositivo principale. Questo oggetto è una mappa che contiene le seguenti coppie chiave-valore: `ComponentToVersionMap`

`key`

Il nome del componente.

`value`

La versione del componente.

`rootComponentsToRemove`(Python:) `root_components_to_remove`

(Facoltativo) I componenti da disinstallare dal dispositivo principale. Specificate un elenco in cui ogni voce è il nome di un componente.

`componentToConfiguration`(Python:) `component_to_configuration`

(Facoltativo) La configurazione viene aggiornata per ogni componente della distribuzione. Questo oggetto è una mappa che contiene le seguenti coppie chiave-valore: `ComponentToConfiguration`

`key`

Il nome del componente.

`value`

L'oggetto JSON di aggiornamento della configurazione per il componente. L'oggetto JSON deve avere il seguente formato.

```
{
  "MERGE": {
    "config-key": "config-value"
  },
  "RESET": [
    "path/to/reset/"
  ]
}
```

Per ulteriori informazioni sugli aggiornamenti della configurazione, vedereAggiornamento delle configurazioni dei componenti.

`componentToRunWithInfo`(Python:) `component_to_run_with_info`

(Facoltativo) La configurazione di runtime per ogni componente della distribuzione. Questa configurazione include l'utente di sistema che possiede i processi di ciascun componente e i limiti

di sistema da applicare a ciascun componente. Questo oggetto è una mappa che contiene le seguenti coppie chiave-valore: `ComponentToRunWithInfo`

`key`

Il nome del componente.

`value`

La configurazione di runtime per il componente. Se si omette un parametro di configurazione di runtime, il software AWS IoT Greengrass Core utilizza i valori predefiniti configurati sul nucleo Greengrass. Questo oggetto contiene `RunWithInfo` le seguenti informazioni:

`posixUser`(Python:) `posix_user`

(Facoltativo) L'utente e, facoltativamente, il gruppo del sistema POSIX da utilizzare per eseguire questo componente sui dispositivi core di Linux. L'utente e il gruppo, se specificato, devono esistere su ogni dispositivo principale di Linux. Specifica l'utente e il gruppo separati da due punti (`:`) nel seguente formato: `user:group`. Il gruppo è facoltativo. Se non si specifica un gruppo, il software AWS IoT Greengrass Core utilizza il gruppo primario per l'utente. Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti.

`windowsUser`(Python:) `windows_user`

(Facoltativo) L'utente Windows da utilizzare per eseguire questo componente sui dispositivi Windows principali. L'utente deve esistere su ogni dispositivo Windows principale e il nome e la password devono essere memorizzati nell'istanza di Credentials Manager dell' LocalSystem account. Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti.

`systemResourceLimits`(Python:) `system_resource_limits`

(Facoltativo) I limiti delle risorse di sistema da applicare ai processi di questo componente. È possibile applicare limiti di risorse di sistema a componenti Lambda generici e non containerizzati. Per ulteriori informazioni, consulta Configura i limiti delle risorse di sistema per i componenti.

AWS IoT Greengrassattualmente non supporta questa funzionalità sui dispositivi Windows principali.

Questo oggetto contiene `SystemResourceLimits` le seguenti informazioni:

`cpus`

(Facoltativo) La quantità massima di tempo di CPU che i processi di questo componente possono utilizzare sul dispositivo principale. Il tempo totale della CPU di un dispositivo principale è equivalente al numero di core CPU del dispositivo. Ad esempio, su un dispositivo principale con 4 core CPU, è possibile impostare questo valore in modo da 2 limitare i processi di questo componente al 50% di utilizzo di ciascun core della CPU. Su un dispositivo con 1 core di CPU, puoi impostare questo valore `0.25` per limitare i processi di questo componente al 25 percento di utilizzo della CPU. Se imposti questo valore su un numero maggiore del numero di core della CPU, il software AWS IoT Greengrass Core non limita l'utilizzo della CPU del componente.

`memory`

(Facoltativo) La quantità massima di RAM (in kilobyte) che i processi di questo componente possono utilizzare sul dispositivo principale.

`groupName`(Python:) `group_name`

(Facoltativo) Il nome del gruppo di oggetti a cui indirizzare questa distribuzione.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`deploymentId`(Python:) `deployment_id`

L'ID della distribuzione locale creata dalla richiesta.

ListLocalDeployments

Ottiene lo stato delle ultime 10 distribuzioni locali.

Questa operazione fornisce le stesse funzionalità del comando deployment list nella CLI di Greengrass.

Richiesta

La richiesta di questa operazione non ha parametri.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`localDeployments`(Python:) `local_deployments`

L'elenco delle distribuzioni locali. Ogni oggetto in questo elenco è un `LocalDeployment` oggetto che contiene le seguenti informazioni:

`deploymentId`(Python:) `deployment_id`

L'ID della distribuzione locale.

`status`

Lo stato della distribuzione locale. Questo enum`DeploymentStatus`, ha i seguenti valori:

- `QUEUED`
- `IN_PROGRESS`
- `SUCCEEDED`
- `FAILED`

GetLocalDeploymentStatus

Ottiene lo stato di una distribuzione locale.

Questa operazione fornisce le stesse funzionalità del comando deployment status nella CLI di Greengrass.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`deploymentId`(Python:) `deployment_id`

L'ID della distribuzione locale da ottenere.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`deployment`

La distribuzione locale. Questo oggetto contiene `LocalDeployment` le seguenti informazioni:

`deploymentId`(Python:) `deployment_id`

L'ID della distribuzione locale.

`status`

Lo stato della distribuzione locale. Questo enum`DeploymentStatus`, ha i seguenti valori:

- `QUEUED`
- `IN_PROGRESS`
- `SUCCEEDED`
- `FAILED`

ListComponents

Ottiene il nome, la versione, lo stato e la configurazione di ogni componente principale sul dispositivo principale. Un componente root è un componente specificato in una distribuzione. Questa risposta non include i componenti che vengono installati come dipendenze di altri componenti.

Questa operazione fornisce le stesse funzionalità del comando dell'elenco dei componenti nella CLI di Greengrass.

Richiesta

La richiesta di questa operazione non ha parametri.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`components`

L'elenco dei componenti principali del dispositivo principale. Ogni oggetto in questo elenco è un `ComponentDetails` oggetto che contiene le seguenti informazioni:

`componentName`(Python:) `component_name`

Il nome del componente.

`version`

La versione del componente.

`state`

Lo stato del componente. Questo stato può essere uno dei seguenti:

- `BROKEN`
- `ERRORED`
- `FINISHED`
- `INSTALLED`
- `NEW`
- `RUNNING`
- `STARTING`
- `STOPPING`

`configuration`

La configurazione del componente come oggetto JSON.

GetComponentDetails

Ottiene la versione, lo stato e la configurazione di un componente sul dispositivo principale.

Questa operazione fornisce le stesse funzionalità del comando component details nella CLI di Greengrass.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`componentName`(Python:) `component_name`

Il nome del componente da ottenere.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`componentDetails`(Python:) `component_details`

I dettagli del componente. Questo oggetto contiene `ComponentDetails` le seguenti informazioni:

`componentName`(Python:) `component_name`

Il nome del componente.

`version`

La versione del componente.

`state`

Lo stato del componente. Questo stato può essere uno dei seguenti:

- `BROKEN`
- `ERRORED`
- `FINISHED`
- `INSTALLED`
- `NEW`
- `RUNNING`
- `STARTING`
- `STOPPING`

`configuration`

La configurazione del componente come oggetto JSON.

RestartComponent

Riavvia un componente sul dispositivo principale.

> **Note**
>
> Sebbene sia possibile riavviare qualsiasi componente, si consiglia di riavviare solo i componenti generici.

Questa operazione fornisce le stesse funzionalità del comando di riavvio del componente nella CLI di Greengrass.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`componentName`(Python:) `component_name`

Il nome del componente.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`restartStatus`(Python:) `restart_status`

Lo stato della richiesta di riavvio. Lo stato della richiesta può essere uno dei seguenti:

- `SUCCEEDED`
- `FAILED`

`message`

Un messaggio sul motivo per cui il componente non è riuscito a riavviarsi, se la richiesta non è riuscita.

StopComponent

Interrompe i processi di un componente sul dispositivo principale.

> **Note**
>
> Sebbene sia possibile interrompere qualsiasi componente, si consiglia di interrompere solo i componenti generici.

Questa operazione fornisce le stesse funzionalità del comando component stop nella CLI di Greengrass.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`componentName`(Python:) `component_name`

Il nome del componente.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`stopStatus`(Python:) `stop_status`

Lo stato della richiesta di interruzione. Lo stato della richiesta può essere uno dei seguenti:

- `SUCCEEDED`
- `FAILED`

`message`

Un messaggio sul motivo per cui il componente non è riuscito a fermarsi, se la richiesta non è riuscita.

CreateDebugPassword

Genera una password casuale che è possibile utilizzare per accedere al componente della console di debug locale. La password scade 8 ore dopo la generazione.

Questa operazione fornisce le stesse funzionalità del get-debug-password comando nella CLI di Greengrass.

Richiesta

La richiesta di questa operazione non ha parametri.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`username`

Il nome utente da usare per accedere.

`password`

La password da usare per accedere.

`passwordExpiration`(Python:) `password_expiration`

L'ora in cui scade la password.

`certificateSHA256Hash`(Python:) `certificate_sha256_hash`

L'impronta digitale SHA-256 per il certificato autofirmato utilizzato dalla console di debug locale quando HTTPS è abilitato. Quando apri la console di debug locale, usa questa impronta digitale per verificare che il certificato sia legittimo e che la connessione sia sicura.

`certificateSHA1Hash`(Python:) `certificate_sha1_hash`

L'impronta digitale SHA-1 per il certificato autofirmato che la console di debug locale utilizza quando HTTPS è abilitato. Quando apri la console di debug locale, usa questa impronta digitale per verificare che il certificato sia legittimo e che la connessione sia sicura.

Autentica e autorizza i dispositivi client

> **Note**
>
> Questa funzionalità è disponibile per la versione 2.6.0 e successive del componente Greengrass nucleus.

Utilizza il servizio IPC di autenticazione dei dispositivi client per sviluppare un componente broker locale personalizzato a cui possano connettersi i dispositivi IoT locali, come i dispositivi client.

Per utilizzare queste operazioni IPC, includi la versione 2.2.0 o successiva del componente di autenticazione del dispositivo client come dipendenza nel componente personalizzato. È quindi possibile utilizzare le operazioni IPC nei componenti personalizzati per effettuare le seguenti operazioni:

- Verifica l'identità dei dispositivi client che si connettono al dispositivo principale.
- Crea una sessione per consentire a un dispositivo client di connettersi al dispositivo principale.
- Verifica se un dispositivo client è autorizzato a eseguire un'azione.
- Ricevi una notifica quando il certificato del server del dispositivo principale cambia.

Argomenti

- Versioni SDK minime
- Autorizzazione
- VerifyClientDeviceIdentity
- GetClientDeviceAuthToken
- AuthorizeClientDeviceAction
- SubscribeToCertificateUpdates

Versioni SDK minime

La tabella seguente elenca le versioni minime da utilizzare per interagire con il servizio IPC di autenticazione del dispositivo client. SDK per dispositivi AWS IoT

SDK	Versione minima
SDK per dispositivi AWS IoTper Java v2	v1.9.3
SDK per dispositivi AWS IoTper Python v2	v1.11.3
SDK per dispositivi AWS IoTper C++ v2	v1.18.3
SDK per dispositivi AWS IoTper v2 JavaScript	v1.12.0

Autorizzazione

Per utilizzare il servizio IPC di autenticazione del dispositivo client in un componente personalizzato, è necessario definire politiche di autorizzazione che consentano al componente di eseguire queste operazioni. Per informazioni sulla definizione delle politiche di autorizzazione, vedere. Autorizza i componenti a eseguire operazioni IPC

Le politiche di autorizzazione per l'autenticazione e l'autorizzazione dei dispositivi client hanno le seguenti proprietà.

Identificatore del servizio IPC: `aws.greengrass.clientdevices.Auth`

Operazione	Descrizione	Risorse
`aws.greengrass#VerifyClientDeviceIdentity`	Consente a un componente di verificare l'identità di un dispositivo client.	*
`aws.greengrass#GetClientDeviceAuthToken`	Consente a un componente di convalidare le credenziali di un dispositivo client e di creare una sessione per quel dispositivo client.	*
`aws.greengrass#AuthorizeClientDeviceAction`	Consente a un componente di verificare se un dispositivo client è autorizzato a eseguire un'azione.	*
`aws.greengrass#SubscribeToCertificateUpdates`	Consente a un componente di ricevere notifiche quando il certificato del server del dispositivo principale ruota.	*
*	Consente a un componente di eseguire tutte le operazioni del servizio IPC di autenticazione del dispositivo client.	*

Esempi di politiche di autorizzazione

È possibile fare riferimento al seguente esempio di politica di autorizzazione per configurare le politiche di autorizzazione per i componenti.

Example Esempio di politica di autorizzazione

Il seguente esempio di politica di autorizzazione consente a un componente di eseguire tutte le operazioni IPC di autenticazione dei dispositivi client.

```
{
  "accessControl": {
    "aws.greengrass.clientdevices.Auth": {
      "com.example.MyLocalBrokerComponent:clientdevices:1": {
        "policyDescription": "Allows access to authenticate and authorize client
 devices.",
        "operations": [
          "aws.greengrass#VerifyClientDeviceIdentity",
          "aws.greengrass#GetClientDeviceAuthToken",
          "aws.greengrass#AuthorizeClientDeviceAction",
          "aws.greengrass#SubscribeToCertificateUpdates"
        ],
        "resources": [
          "*"
        ]
      }
    }
  }
}
```

VerifyClientDeviceIdentity

Verifica l'identità di un dispositivo client. Questa operazione verifica se il dispositivo client è validoAWS IoT.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`credential`

Le credenziali del dispositivo client. Questo oggetto contiene `ClientDeviceCredential` le seguenti informazioni:

`clientDeviceCertificate`(Python:) `client_device_certificate`

Il certificato del dispositivo X.509 del dispositivo client.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`isValidClientDevice`(Python:) `is_valid_client_device`

Se l'identità del dispositivo client è valida.

GetClientDeviceAuthToken

Convalida le credenziali di un dispositivo client e crea una sessione per il dispositivo client. Questa operazione restituisce un token di sessione che è possibile utilizzare nelle richieste successive per autorizzare le azioni del dispositivo client.

Per connettere correttamente un dispositivo client, il componente di autenticazione del dispositivo client deve concedere l'`mqtt:connect`autorizzazione per l'ID client utilizzato dal dispositivo client.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`credential`

Le credenziali del dispositivo client. Questo oggetto contiene `CredentialDocument` le seguenti informazioni:

`mqttCredential`(Python:) `mqtt_credential`

Le credenziali MQTT del dispositivo client. Specificare l'ID client e il certificato utilizzati dal dispositivo client per la connessione. Questo oggetto contiene `MQTTCredential` le seguenti informazioni:

`clientId`(Python:) `client_id`

L'ID client da utilizzare per la connessione.

`certificatePem`(Python:) `certificate_pem`

Il certificato del dispositivo X.509 da utilizzare per la connessione.

`username`

> **Note**
>
> Questa proprietà non è attualmente utilizzata.

`password`

> **Note**
>
> Questa proprietà non è attualmente utilizzata.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`clientDeviceAuthToken`(Python:) `client_device_auth_token`

Il token di sessione per il dispositivo client. È possibile utilizzare questo token di sessione nelle richieste successive per autorizzare le azioni di questo dispositivo client.

AuthorizeClientDeviceAction

Verifica se un dispositivo client è autorizzato a eseguire un'azione su una risorsa. Le politiche di autorizzazione dei dispositivi client specificano le autorizzazioni che i dispositivi client possono eseguire mentre sono connessi a un dispositivo principale. Le politiche di autorizzazione dei dispositivi client vengono definite quando si configura il componente di autenticazione del dispositivo client.

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`clientDeviceAuthToken`(Python:) `client_device_auth_token`

Il token di sessione per il dispositivo client.

`operation`

L'operazione da autorizzare.

`resource`

La risorsa in cui il dispositivo client esegue l'operazione.

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`isAuthorized`(Python:) `is_authorized`

Se il dispositivo client è autorizzato a eseguire l'operazione sulla risorsa.

SubscribeToCertificateUpdates

Abbonati per ricevere il nuovo certificato server del dispositivo principale ogni volta che ruota. Quando il certificato del server cambia, i broker devono ricaricarlo utilizzando il nuovo certificato del server.

Per impostazione predefinita, il componente di autenticazione del dispositivo client ruota i certificati del server ogni 7 giorni. È possibile configurare l'intervallo di rotazione tra 2 e 10 giorni.

Questa operazione è un'operazione di sottoscrizione in cui ci si iscrive a un flusso di messaggi di eventi. Per utilizzare questa operazione, definite un gestore di risposte di flusso con funzioni che gestiscono i messaggi di evento, gli errori e la chiusura dei flussi. Per ulteriori informazioni, consulta Iscriviti ai flussi di eventi IPC.

Tipo di messaggio di evento: `CertificateUpdateEvent`

Richiesta

La richiesta di questa operazione ha i seguenti parametri:

`certificateOptions`(Python:) `certificate_options`

I tipi di aggiornamenti dei certificati a cui sottoscrivere. Questo oggetto contiene `CertificateOptions` le seguenti informazioni:

`certificateType`(Python:) `certificate_type`

Il tipo di aggiornamenti dei certificati a cui sottoscrivere. Scegli la seguente opzione:

- `SERVER`

Risposta

La risposta di questa operazione contiene le seguenti informazioni:

`messages`

Il flusso di messaggi. Questo oggetto contiene `CertificateUpdateEvent` le seguenti informazioni:

`certificateUpdate`(Python:) `certificate_update`

Le informazioni sul nuovo certificato. Questo oggetto contiene `CertificateUpdate` le seguenti informazioni:

`certificate`

Il certificato.

`privateKey`(Python:) `private_key`

La chiave privata del certificato.

`publicKey`(Python:) `public_key`

La chiave pubblica del certificato.

`caCertificates`(Python:) `ca_certificates`

L'elenco dei certificati di autorità di certificazione (CA) nella catena di certificati CA del certificato.

Interagisci con dispositivi IoT locali

I dispositivi client sono dispositivi IoT locali che si connettono e comunicano con un dispositivo core Greengrass tramite MQTT. È possibile connettere i dispositivi client ai dispositivi principali per effettuare le seguenti operazioni:

- Interagisci con i messaggi MQTT nei componenti Greengrass.
- Inoltra messaggi e dati tra dispositivi client e. AWS IoT Core
- Interagisci con le ombre dei dispositivi client nei componenti Greengrass.
- Sincronizza le ombre dei dispositivi client con. AWS IoT Core

Per connettersi a un dispositivo principale, i dispositivi client possono utilizzare il cloud discovery. I dispositivi client si connettono al servizio AWS IoT Greengrass cloud per recuperare informazioni sui dispositivi principali a cui possono connettersi. Quindi, possono connettersi a un dispositivo principale per elaborare i messaggi e sincronizzare i dati con il servizio AWS IoT Core cloud.

Puoi seguire un tutorial che spiega come configurare un dispositivo principale per connettersi e comunicare con qualsiasi AWS IoT cosa. Questo tutorial esplora anche come sviluppare un componente Greengrass personalizzato che interagisca con i dispositivi client. Per ulteriori informazioni, consulta Tutorial: Interagisci con i dispositivi IoT locali tramite MQTT.

Argomenti

AWS-componenti del dispositivo client forniti

AWS IoT Greengrassfornisce i seguenti componenti pubblici che è possibile distribuire sui dispositivi principali. Questi componenti consentono ai dispositivi client di connettersi e comunicare con un dispositivo principale.

> **Note**
>
> Diversi componenti AWS forniti dipendono da versioni minori specifiche del nucleo Greengrass. A causa di questa dipendenza, è necessario aggiornare questi componenti quando si aggiorna il nucleo di Greengrass a una nuova versione secondaria. Per informazioni sulle versioni specifiche del nucleo da cui dipende ogni componente, consultate l'argomento relativo ai componenti. Per ulteriori informazioni sull'aggiornamento del nucleo, vedere. Aggiornamento del software AWS IoT Greengrass Core (OTA)

Quando un componente ha un tipo di componente sia generico che Lambda, la versione corrente del componente è di tipo generico e una versione precedente del componente è di tipo Lambda.

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Autenticazione del dispositivo client	Consente ai dispositivi IoT locali, chiamati dispositivi client, di connettersi al dispositivo principale.	Plug-in	Linux, Windows	Sì
Rilevatore IP	Riporta le informazioni sulla connettività del broker MQTT aAWS IoT Greengrass, in modo che i dispositivi client possano scoprire come connettersi.	Plug-in	Linux, Windows	Sì

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Ponte MQTT	Inoltra messaggi MQTT tra dispositivi client, AWS IoT Greengrass pubblicazione/sottoscrizione locali e. AWS IoT Core	Plug-in	Linux, Windows	Sì
Broker MQTT 3.1.1 (Moquette)	Esegue un broker MQTT 3.1.1 che gestisce i messaggi tra i dispositivi client e il dispositivo principale.	Plug-in	Linux, Windows	Sì
Broker MQTT 5 (EMQX)	Esegue un broker MQTT 5 che gestisce i messaggi tra i dispositivi client e il dispositivo principale.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Gestore delle ombre	Consente l'interazione con le ombre sul dispositivo principale. Gestisce l'archiviazione dei documenti shadow e anche la sincronizzazione degli stati shadow locali con il servizio AWS IoT Device Shadow.	Plug-in	Linux, Windows	Sì

Connect i dispositivi client ai dispositivi principali

Puoi configurare il cloud discovery per connettere i dispositivi client ai dispositivi principali. Quando configuri il cloud discovery, i dispositivi client possono connettersi al servizio AWS IoT Greengrass cloud per recuperare informazioni sui dispositivi principali a cui possono connettersi. Quindi, i dispositivi client possono tentare di connettersi a ciascun dispositivo principale fino a quando non si connettono correttamente.

Per utilizzare cloud discovery, devi fare quanto segue:

- Associa i dispositivi client ai dispositivi principali a cui possono connettersi.
- Specificate gli endpoint del broker MQTT a cui i dispositivi client possono connettersi a ciascun dispositivo principale.
- Implementa i componenti sul dispositivo principale che abilitano il supporto per i dispositivi client.

Puoi anche distribuire componenti opzionali per effettuare le seguenti operazioni:

- Inoltra i messaggi tra i dispositivi client, i componenti Greengrass e AWS IoT Core il servizio cloud.
- Gestisci automaticamente gli endpoint del broker MQTT dei dispositivi principali per te.
- Gestisci le ombre dei dispositivi client locali e sincronizza le ombre con il servizio cloud. AWS IoT Core

È inoltre necessario rivedere e aggiornare la AWS IoT politica del dispositivo principale per verificare che disponga delle autorizzazioni necessarie per connettere i dispositivi client. Per ulteriori informazioni, consulta Requisiti.

Dopo aver configurato il cloud discovery, puoi testare le comunicazioni tra un dispositivo client e un dispositivo principale. Per ulteriori informazioni, consulta Verifica le comunicazioni con i dispositivi client.

Argomenti

Requisiti

Per connettere i dispositivi client a un dispositivo principale, è necessario disporre di quanto segue:

- Il dispositivo principale deve eseguire Greengrass nucleus v2.2.0 o versione successiva.

- Il ruolo di servizio Greengrass associato all'AWS IoT Greengrassutente Account AWS nella AWS regione in cui opera il dispositivo principale. Per ulteriori informazioni, consulta Configurazione del ruolo del servizio Greengrass.
- La AWS IoT politica del dispositivo principale deve consentire le seguenti autorizzazioni:
  - `greengrass:PutCertificateAuthorities`
  - `greengrass:VerifyClientDeviceIdentity`
  - `greengrass:VerifyClientDeviceIoTCertificateAssociation`
  - `greengrass:GetConnectivityInfo`
  - `greengrass:UpdateConnectivityInfo`— (Facoltativo) Questa autorizzazione è necessaria per utilizzare il componente del rilevatore IP, che riporta le informazioni sulla connettività di rete del dispositivo principale al AWS IoT Greengrass servizio cloud.
  - `iot:GetThingShadowiot:UpdateThingShadow`, e `iot:DeleteThingShadow` — (Facoltativo) Queste autorizzazioni sono necessarie per utilizzare il componente shadow manager con cui sincronizzare le ombre dei dispositivi client. AWS IoT Core Questa funzionalità richiede Greengrass nucleus v2.6.0 o successivo, shadow manager v2.2.0 o successivo e MQTT bridge v2.2.0 o successivo.

  Per ulteriori informazioni, consulta Configura la policy relativa agli AWS IoT oggetti.

  > **Note**
  >
  > Se hai utilizzato la AWS IoT politica predefinita quando hai installato il software AWS IoT Greengrass Core, il dispositivo principale dispone di una AWS IoT politica che consente l'accesso a tutte le azioni (). AWS IoT Greengrass `greengrass:*`

- AWS IoTcose che puoi connettere come dispositivi client. Per ulteriori informazioni, consulta AWS IoTCreate resources nella AWS IoT CoreDeveloper Guide.
- Il dispositivo client deve connettersi utilizzando un ID client. Un ID client è il nome di un oggetto. Non verrà accettato nessun altro ID cliente.
- La AWS IoT politica di ogni dispositivo client deve consentire l'`greengrass:Discover`autorizzazione. Per ulteriori informazioni, consulta AWS IoTPolicy minima per i dispositivi client.

Argomenti

- Configurazione del ruolo del servizio Greengrass

- Configura la policy relativa agli AWS IoT oggetti

Configurazione del ruolo del servizio Greengrass

Il ruolo di servizio Greengrass è un ruolo di servizio AWS Identity and Access Management (IAM) che AWS IoT Greengrass autorizza l'accesso alle risorse AWS dei servizi per conto dell'utente. Questo ruolo consente di AWS IoT Greengrass verificare l'identità dei dispositivi client e gestire le informazioni principali sulla connettività dei dispositivi.

Se non hai precedentemente impostato il ruolo di servizio Greengrass in questa regione, devi associare un ruolo di servizio Greengrass a un ruolo di servizio Greengrass AWS IoT Greengrass per te Account AWS in questa regione.

Quando utilizzi la pagina Configure core device discovery nella AWS IoT Greengrassconsole, AWS IoT Greengrass configura automaticamente il ruolo del servizio Greengrass. Altrimenti, puoi configurarlo manualmente utilizzando la AWS IoTconsole o l'AWS IoT GreengrassAPI.

In questa sezione, si verifica se il ruolo di servizio Greengrass è impostato. Se non è configurato, crei un nuovo ruolo di servizio Greengrass a cui AWS IoT Greengrass associarti Account AWS in questa regione.

Configurazione del ruolo di servizio Greengrass (console)

1. Verifica se il ruolo di servizio Greengrass è associato AWS IoT Greengrass al tuo ruolo Account AWS in questa regione. Esegui questa operazione:

   a. Passare alla console AWS IoT.
   b. Nel pannello di navigazione scegli Impostazioni.
   c. Nella sezione Ruolo di servizio Greengrass, trova Ruolo di servizio corrente per vedere se è associato un ruolo di servizio Greengrass.

      Se hai un ruolo di servizio Greengrass associato, soddisfi questo requisito per utilizzare il componente del rilevatore IP. Passa a Configura la policy relativa agli AWS IoT oggetti.
2. Se il ruolo di servizio Greengrass non è associato AWS IoT Greengrass al tuo ruolo Account AWS in questa regione, crea un ruolo di servizio Greengrass e associalo. Esegui questa operazione:

   a. Passare alla IAM console (Console IAM).
   b. Scegliere Roles (Ruoli).

c. Scegli Crea ruolo.

d. Nella pagina Crea ruolo, procedi come segue:

   i. In Tipo di entità affidabile, scegli Servizio AWS.

   ii. In Caso d'uso, Casi d'uso per altro Servizi AWS, scegli Greengrass, seleziona Greengrass. Questa opzione specifica l'aggiunta AWS IoT Greengrass come entità attendibile che può assumere questo ruolo.

   iii. Seleziona Avanti.

   iv. In Politiche di autorizzazione, seleziona AWSGreengrassResourceAccessRolePolicyda associare al ruolo.

   v. Seleziona Avanti.

   vi. In Nome ruolo, inserisci un nome per il ruolo, ad esempio**`Greengrass_ServiceRole`**.

   vii. Scegli Crea ruolo.

e. Passare alla console AWS IoT.

f. Nel pannello di navigazione scegli Impostazioni.

g. Nella sezione Ruolo di servizio Greengrass, scegli Collega ruolo.

h. Nella modalità Update Greengrass service role, seleziona il ruolo IAM che hai creato, quindi scegli Attach role.

Configurare il ruolo del servizio Greengrass () AWS CLI

1. Verifica se il ruolo di servizio Greengrass è associato AWS IoT Greengrass al tuo ruolo Account AWS in questa regione.

   ```
   aws greengrassv2 get-service-role-for-account
   ```

   Se il ruolo di servizio Greengrass è associato, l'operazione restituisce una risposta contenente informazioni sul ruolo.

   Se hai un ruolo di servizio Greengrass associato, soddisfi questo requisito per utilizzare il componente del rilevatore IP. Passa a Configura la policy relativa agli AWS IoT oggetti.

2. Se il ruolo di servizio Greengrass non è associato AWS IoT Greengrass al tuo ruolo Account AWS in questa regione, crea un ruolo di servizio Greengrass e associalo. Esegui questa operazione:

a. Creare un ruolo con una policy di attendibilità che consenta a AWS IoT Greengrass di assumere tale ruolo. In questo esempio viene creato un ruolo denominato `Greengrass_ServiceRole`, ma è possibile utilizzare un nome diverso. Ti consigliamo di includere anche le chiavi di contesto della `aws:SourceArn` condizione `aws:SourceAccount` globale nella tua politica di fiducia per evitare il confuso problema della sicurezza dei vicedirettori. Le chiavi di contesto delle condizioni limitano l'accesso per consentire solo le richieste che provengono dall'account specificato e dall'area di lavoro Greengrass. Per ulteriori informazioni sul problema del "confused deputy", consulta Prevenzione del problema "confused deputy" tra servizi.

   Linux or Unix

   ```
   aws iam create-role --role-name Greengrass_ServiceRole --assume-role-policy-
   document '{
     "Version": "2012-10-17",
     "Statement": [
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "greengrass.amazonaws.com"
         },
         "Action": "sts:AssumeRole",
         "Condition": {
           "ArnLike": {
             "aws:SourceArn": "arn:aws:greengrass:region:account-id:*"
           },
           "StringEquals": {
             "aws:SourceAccount": "account-id"
           }
         }
       }
     ]
   }'
   ```

   Windows Command Prompt (CMD)

   ```
   aws iam create-role --role-name Greengrass_ServiceRole --assume-role-policy-
   document "{\\"Version\\":\\"2012-10-17\\",\\"Statement\\":[{\\"Effect\
   \":\\"Allow\\",\\"Principal\\":{\\"Service\\":\\"greengrass.amazonaws.com
   \\"},\\"Action\\":\\"sts:AssumeRole\\",\\"Condition\\":{\\"ArnLike\\":
   ```

```
{\\"aws:SourceArn\\":\\"arn:aws:greengrass:region:account-id:*\\"},\
\"StringEquals\\":{\\"aws:SourceAccount\\":\\"account-id\\"}}}]}"
```

PowerShell

```
aws iam create-role --role-name Greengrass_ServiceRole --assume-role-policy-
document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "greengrass.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:greengrass:region:account-id:*"
        },
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        }
      }
    }
  ]
}'
```

b. Copiare il ruolo ARN dai metadati del ruolo nell'output. Utilizzare l'ARN per associare un ruolo all'account.

c. Collegare la policy `AWSGreengrassResourceAccessRolePolicy` al ruolo.

```
aws iam attach-role-policy --role-name Greengrass_ServiceRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSGreengrassResourceAccessRolePolicy
```

d. Associa il ruolo di servizio Greengrass a AWS IoT Greengrass for your. Account AWS Sostituisci *`role-arn`* con l'ARN del ruolo di servizio.

```
aws greengrassv2 associate-service-role-to-account --role-arn role-arn
```

L'operazione restituisce la seguente risposta se ha esito positivo.

```
{
  "associatedAt": "timestamp"
}
```

Configura la policy relativa agli AWS IoT oggetti

I dispositivi principali utilizzano i certificati dei dispositivi X.509 per autorizzare le connessioni a. AWS Si allegano AWS IoT politiche ai certificati dei dispositivi per definire le autorizzazioni per un dispositivo principale. Per ulteriori informazioni, consultare Policy AWS IoT per operazioni del piano dei dati e AWS IoTPolitica minima per supportare i dispositivi client.

Per connettere i dispositivi client a un dispositivo principale, la AWS IoT politica del dispositivo principale deve consentire le seguenti autorizzazioni:

- `greengrass:PutCertificateAuthorities`
- `greengrass:VerifyClientDeviceIdentity`
- `greengrass:VerifyClientDeviceIoTCertificateAssociation`
- `greengrass:GetConnectivityInfo`
- `greengrass:UpdateConnectivityInfo`— (Facoltativo) Questa autorizzazione è necessaria per utilizzare il componente del rilevatore IP, che riporta le informazioni sulla connettività di rete del dispositivo principale al AWS IoT Greengrass servizio cloud.
- `iot:GetThingShadow`iot:UpdateThingShadow, e `iot:DeleteThingShadow` — (Facoltativo) Queste autorizzazioni sono necessarie per utilizzare il componente shadow manager con cui sincronizzare le ombre dei dispositivi client. AWS IoT Core Questa funzionalità richiede Greengrass nucleus v2.6.0 o successivo, shadow manager v2.2.0 o successivo e MQTT bridge v2.2.0 o successivo.

In questa sezione, esaminate le AWS IoT politiche per il vostro dispositivo principale e aggiungete le eventuali autorizzazioni necessarie mancanti. Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per il provisioning delle risorse, il tuo dispositivo principale dispone di una AWS IoT politica che consente l'accesso a tutte le AWS IoT Greengrass azioni ()`greengrass:*`. In questo caso, è necessario aggiornare la AWS IoT policy solo se si prevede di distribuire il componente shadow manager con cui sincronizzare le ombre dei dispositivi. AWS IoT Core Altrimenti, puoi saltare questa sezione.

Configura la policy AWS IoT relativa agli oggetti (console)

1. Nel menu di navigazione della AWS IoT Greengrassconsole, scegli Dispositivi principali.
2. Nella pagina Dispositivi principali, scegli il dispositivo principale da aggiornare.
3. Nella pagina dei dettagli del dispositivo principale, scegli il link all'oggetto del dispositivo principale. Questo link apre la pagina dei dettagli dell'oggetto nella AWS IoT console.
4. Nella pagina dei dettagli dell'oggetto, scegli Certificati.
5. Nella scheda Certificati, scegli il certificato attivo dell'oggetto.
6. Nella pagina dei dettagli del certificato, scegli Politiche.
7. Nella scheda Politiche, scegli la AWS IoT politica da rivedere e aggiornare. Puoi aggiungere le autorizzazioni richieste a qualsiasi policy allegata al certificato attivo del dispositivo principale.

   > **Note**
   >
   > Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per il provisioning delle risorse, hai due AWS IoT politiche. Ti consigliamo di scegliere la politica denominata GreengrassV2IoTThingPolicy, se esiste. I dispositivi principali creati con il programma di installazione rapida utilizzano questo nome di policy per impostazione predefinita. Se aggiungi autorizzazioni a questo criterio, concedi tali autorizzazioni anche ad altri dispositivi principali che utilizzano questo criterio.

8. Nella panoramica della politica, scegli Modifica versione attiva.
9. Rivedi la politica per le autorizzazioni richieste e aggiungi le autorizzazioni richieste mancanti.

   - `greengrass:PutCertificateAuthorities`
   - `greengrass:VerifyClientDeviceIdentity`
   - `greengrass:VerifyClientDeviceIoTCertificateAssociation`
   - `greengrass:GetConnectivityInfo`
   - `greengrass:UpdateConnectivityInfo`— (Facoltativo) Questa autorizzazione è necessaria per utilizzare il componente IP Detector, che riporta le informazioni di connettività di rete del dispositivo principale al AWS IoT Greengrass servizio cloud.
   - `iot:GetThingShadow`iot:UpdateThingShadow, e `iot:DeleteThingShadow` — (Facoltativo) Queste autorizzazioni sono necessarie per utilizzare il componente shadow manager con cui sincronizzare le ombre dei dispositivi client. AWS IoT Core Questa

funzionalità richiede Greengrass nucleus v2.6.0 o successivo, shadow manager v2.2.0 o successivo e MQTT bridge v2.2.0 o successivo.

10. (Facoltativo) Per consentire al dispositivo principale di sincronizzare gli shadows con, aggiungete la seguente dichiarazione alla policy. AWS IoT Core Se prevedi di interagire con le ombre dei dispositivi client, ma non di sincronizzarle con loroAWS IoT Core, salta questo passaggio. Sostituisci la *regione* e l'*ID dell'account* con la regione che usi e il tuo numero. Account AWS

    - Questa istruzione di esempio consente l'accesso alle ombre del dispositivo di tutti gli oggetti. Per seguire le migliori pratiche di sicurezza, è possibile limitare l'accesso solo al dispositivo principale e ai dispositivi client collegati al dispositivo principale. Per ulteriori informazioni, consulta AWS IoTPolitica minima per supportare i dispositivi client.

```
{
  "Effect": "Allow",
  "Action": [
    "iot:GetThingShadow",
    "iot:UpdateThingShadow",
    "iot:DeleteThingShadow"
  ],
  "Resource": [
    "arn:aws:iot:region:account-id:thing/*"
  ]
}
```

Dopo aver aggiunto questa dichiarazione, il documento di policy potrebbe avere un aspetto simile all'esempio seguente.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "greengrass:*"
```

```
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:DeleteThingShadow"
      ],
      "Resource": [
        "arn:aws:iot:region:account-id:thing/*"
      ]
    }
  ]
}
```

11. Per impostare una nuova versione della politica come versione attiva, in Stato della versione della politica, seleziona Imposta la versione modificata come versione attiva per questa politica.
12. Scegli Salva come nuova versione.

Configura la AWS IoT policy relativa agli oggetti (AWS CLI)

1. Elenca i principi fondamentali del AWS IoT dispositivo principale. I principali degli oggetti possono essere certificati di dispositivo X.509 o altri identificatori. Esegui il comando seguente e sostituiscilo *MyGreengrassCore*con il nome del dispositivo principale.

   ```
   aws iot list-thing-principals --thing-name MyGreengrassCore
   ```

   L'operazione restituisce una risposta che elenca i componenti principali del dispositivo principale.

   ```
   {
       "principals": [
           "arn:aws:iot:us-west-2:123456789012:cert/certificateId"
       ]
   }
   ```

2. Identifica il certificato attivo del dispositivo principale. Esegui il comando seguente e sostituisci *CertificateID* con l'ID di ogni certificato del passaggio precedente fino a trovare il certificato

attivo. L'ID del certificato è la stringa esadecimale alla fine dell'ARN del certificato. L'`--query`argomento specifica di visualizzare solo lo stato del certificato.

```
aws iot describe-certificate --certificate-id certificateId --query
 'certificateDescription.status'
```

L'operazione restituisce lo stato del certificato sotto forma di stringa. Ad esempio, se il certificato è attivo, questa operazione genera un output`"ACTIVE"`.

3. Elenca le AWS IoT politiche allegate al certificato. Esegui il comando seguente e sostituisci l'ARN del certificato con l'ARN del certificato.

```
aws iot list-principal-policies --principal arn:aws:iot:us-
west-2:123456789012:cert/certificateId
```

L'operazione restituisce una risposta che elenca le AWS IoT politiche allegate al certificato.

```
{
    "policies": [
        {
            "policyName":
 "GreengrassTESCertificatePolicyMyGreengrassCoreTokenExchangeRoleAlias",
            "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassTESCertificatePolicyMyGreengrassCoreTokenExchangeRoleAlias"
        },
        {
            "policyName": "GreengrassV2IoTThingPolicy",
            "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy"
        }
    ]
}
```

4. Scegli la politica da visualizzare e aggiornare.

> **Note**
>
> Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per il provisioning delle risorse, hai due AWS IoT politiche. Ti consigliamo di scegliere la politica denominata GreengrassV2IoTThingPolicy, se esiste. I dispositivi principali creati con il programma di installazione rapida utilizzano questo nome di policy per

impostazione predefinita. Se aggiungi autorizzazioni a questo criterio, concedi tali autorizzazioni anche ad altri dispositivi principali che utilizzano questo criterio.

5. Scarica il documento della politica. Esegui il comando seguente e sostituisci *GreenGrassV2IoTThingPolicy* con il nome della politica.

```
aws iot get-policy --policy-name GreengrassV2IoTThingPolicy
```

L'operazione restituisce una risposta che contiene il documento della politica e altre informazioni sulla politica. Il documento di policy è un oggetto JSON serializzato come stringa.

```
{
    "policyName": "GreengrassV2IoTThingPolicy",
    "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
    "policyDocument": "{\
  \\"Version\\": \\"2012-10-17\\",\
  \\"Statement\\": [\
    {\
      \\"Effect\\": \\"Allow\\",\
      \\"Action\\": [\
                \\"iot:Connect\\",\
                \\"iot:Publish\\",\
                \\"iot:Subscribe\\",\
                \\"iot:Receive\\",\
                \\"greengrass:*\\"\
],\
      \\"Resource\\": \\"*\\"\
    }\
  ]\
}",
    "defaultVersionId": "1",
    "creationDate": "2021-02-05T16:03:14.098000-08:00",
    "lastModifiedDate": "2021-02-05T16:03:14.098000-08:00",
    "generationId":
 "f19144b798534f52c619d44f771a354f1b957dfa2b850625d9f1d0fde530e75f"
}
```

6. Utilizzate un convertitore online o un altro strumento per convertire la stringa del documento di policy in un oggetto JSON, quindi salvatela in un file denominato. `iot-policy.json`

Ad esempio, se è installato lo strumento jq, è possibile eseguire il comando seguente per ottenere il documento di policy, convertirlo in un oggetto JSON e salvare il documento di policy come oggetto JSON.

```
aws iot get-policy --policy-name GreengrassV2IoTThingPolicy --query
 'policyDocument' | jq fromjson >> iot-policy.json
```

7. Esamina la politica per le autorizzazioni richieste e aggiungi le autorizzazioni richieste mancanti.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per aprire il file.

   ```
   nano iot-policy.json
   ```

   - `greengrass:PutCertificateAuthorities`
   - `greengrass:VerifyClientDeviceIdentity`
   - `greengrass:VerifyClientDeviceIoTCertificateAssociation`
   - `greengrass:GetConnectivityInfo`
   - `greengrass:UpdateConnectivityInfo`— (Facoltativo) Questa autorizzazione è necessaria per utilizzare il componente IP Detector, che riporta le informazioni sulla connettività di rete del dispositivo principale al servizio cloud. AWS IoT Greengrass
   - `iot:GetThingShadow`iot:UpdateThingShadow, e `iot:DeleteThingShadow` — (Facoltativo) Queste autorizzazioni sono necessarie per utilizzare il componente shadow manager con cui sincronizzare le ombre dei dispositivi client. AWS IoT Core Questa funzionalità richiede Greengrass nucleus v2.6.0 o successivo, shadow manager v2.2.0 o successivo e MQTT bridge v2.2.0 o successivo.

8. Salva le modifiche come nuova versione della politica. Esegui il comando seguente e sostituisci *GreenGrassV2IoT ThingPolicy* con il nome della politica.

   ```
   aws iot create-policy-version --policy-name GreengrassV2IoTThingPolicy --policy-
   document file://iot-policy.json --set-as-default
   ```

   L'operazione restituisce una risposta simile all'esempio seguente se ha esito positivo.

   ```
   {
   ```

```
    "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
    "policyDocument": "{\
  \\"Version\\": \\"2012-10-17\\",\
  \\"Statement\\": [\
    {\
      \\"Effect\\": \\"Allow\\",\
      \\"Action\\": [\
\\t\\t\\"iot:Connect\\",\
\\t\\t\\"iot:Publish\\",\
\\t\\t\\"iot:Subscribe\\",\
\\t\\t\\"iot:Receive\\",\
\\t\\t\\"greengrass:*\\"\
      ],\
      \\"Resource\\": \\"*\\"\
    }\
  ]\
}",
    "policyVersionId": "2",
    "isDefaultVersion": true
}
```

Componenti Greengrass per il supporto dei dispositivi client

> **Important**
>
> Il dispositivo principale deve eseguire Greengrass nucleus v2.2.0 o versione successiva per supportare i dispositivi client.

Per consentire ai dispositivi client di connettersi e comunicare con un dispositivo principale, si distribuiscono i seguenti componenti Greengrass sul dispositivo principale:

- Autenticazione del dispositivo client (`aws.greengrass.clientdevices.Auth`)

  Implementa il componente di autenticazione del dispositivo client per autenticare i dispositivi client e autorizzare le azioni dei dispositivi client. Questo componente consente ai tuoi dispositivi di AWS IoT connettersi a un dispositivo principale.

Questo componente richiede alcune configurazioni per utilizzarlo. È necessario specificare i gruppi di dispositivi client e le operazioni che ciascun gruppo è autorizzato a eseguire, ad esempio la connessione e la comunicazione tramite MQTT. Per ulteriori informazioni, consultate Configurazione dei componenti di autenticazione dei dispositivi client.

- Broker MQTT 3.1.1 (Moquette) (`aws.greengrass.clientdevices.mqtt.Moquette`)

  Implementate il componente del broker MQTT Moquette per eseguire un broker MQTT leggero. Il broker Moquette MQTT è conforme a MQTT 3.1.1 e include il supporto locale per QoS 0, QoS 1, QoS 2, messaggi conservati, messaggi di ultima volontà e abbonamenti permanenti.

  Non è necessario configurare questo componente per utilizzarlo. Tuttavia, è possibile configurare la porta su cui questo componente gestisce il broker MQTT. Per impostazione predefinita, utilizza la porta 8883.

- Broker MQTT 5 (EMQX) (`aws.greengrass.clientdevices.mqtt.EMQX`)

  > **Note**
  >
  > Per utilizzare il broker EMQX MQTT 5, è necessario utilizzare Greengrass nucleus v2.6.0 o versione successiva e l'autenticazione del dispositivo client v2.2.0 o successiva.

  Implementate il componente broker EMQX MQTT per utilizzare le funzionalità MQTT 5.0 nella comunicazione tra i dispositivi client e il dispositivo principale. Il broker EMQX MQTT è conforme a MQTT 5.0 e include il supporto per gli intervalli di scadenza delle sessioni e dei messaggi, le proprietà degli utenti, gli abbonamenti condivisi, gli alias degli argomenti e altro ancora.

  Non è necessario configurare questo componente per utilizzarlo. Tuttavia, è possibile configurare la porta su cui questo componente gestisce il broker MQTT. Per impostazione predefinita, utilizza la porta 8883.

- Ponte MQTT (`aws.greengrass.clientdevices.mqtt.Bridge`)

  (Facoltativo) Implementate il componente bridge MQTT per inoltrare messaggi tra dispositivi client (MQTT locale), pubblicazione e sottoscrizione locali e MQTT. AWS IoT Core Configura questo componente per sincronizzare i dispositivi client AWS IoT Core e interagire con i dispositivi client dai componenti Greengrass.

Questo componente richiede una configurazione per essere utilizzato. È necessario specificare le mappature degli argomenti in cui questo componente inoltra i messaggi. Per ulteriori informazioni, vedere Configurazione dei componenti del bridge MQTT.

- Rilevatore IP (`aws.greengrass.clientdevices.IPDetector`)

  (Facoltativo) Implementate il componente del rilevatore IP per segnalare automaticamente gli endpoint del broker MQTT del dispositivo principale al servizio cloud. AWS IoT Greengrass Non è possibile utilizzare questo componente se si dispone di una configurazione di rete complessa, ad esempio una in cui un router inoltra la porta del broker MQTT al dispositivo principale.

  Non è necessario configurare questo componente per utilizzarlo.

- Gestore delle ombre (`aws.greengrass.ShadowManager`)

  > **Note**
  >
  > Per gestire le ombre dei dispositivi client, è necessario utilizzare Greengrass nucleus v2.6.0 o successivo, shadow manager v2.2.0 o successivo e MQTT bridge v2.2.0 o successivo.

  (Facoltativo) Implementate il componente shadow manager per gestire le ombre dei dispositivi client sul dispositivo principale. I componenti Greengrass possono ottenere, aggiornare ed eliminare le ombre dei dispositivi client per interagire con i dispositivi client. È inoltre possibile configurare il componente shadow manager per sincronizzare le ombre dei dispositivi client con il servizio cloud. AWS IoT Core

  Per utilizzare questo componente con le ombre dei dispositivi client, è necessario configurare il componente bridge MQTT per inoltrare i messaggi tra i dispositivi client e lo shadow manager, che utilizza la pubblicazione/sottoscrizione locale. Altrimenti, questo componente non richiede alcuna configurazione per essere utilizzato, ma richiede la configurazione per sincronizzare le ombre del dispositivo.

> **Note**
>
> Si consiglia di implementare solo un componente del broker MQTT. I componenti del bridge MQTT e del rilevatore IP funzionano con un solo componente del broker MQTT alla volta.

Se si distribuiscono più componenti del broker MQTT, è necessario configurarli per utilizzare porte diverse.

Configura cloud discovery (console)

Puoi utilizzare la AWS IoT Greengrass console per associare i dispositivi client, gestire gli endpoint principali dei dispositivi e distribuire componenti per abilitare il supporto dei dispositivi client. Per ulteriori informazioni, consulta Passaggio 2: abilitare il supporto per i dispositivi client.

Configura cloud discovery () AWS CLI

Puoi utilizzare il AWS Command Line Interface (AWS CLI) per associare i dispositivi client, gestire gli endpoint principali dei dispositivi e distribuire componenti per abilitare il supporto dei dispositivi client. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Gestisci le associazioni dei dispositivi client () AWS CLI
- Gestisci gli endpoint principali dei dispositivi
- AWS-componenti del dispositivo client forniti
- Creare distribuzione

Associa i dispositivi client

Per utilizzare il cloud discovery, associa i dispositivi client a un dispositivo principale in modo che possano scoprire il dispositivo principale. Quindi, possono utilizzare l'API Greengrass discovery per recuperare le informazioni di connettività e i certificati per i dispositivi principali associati.

Allo stesso modo, dissociate i dispositivi client da un dispositivo principale per impedire loro di scoprire il dispositivo principale.

Argomenti

- Gestisci le associazioni dei dispositivi client (console)
- Gestisci le associazioni dei dispositivi client () AWS CLI
- Gestisci le associazioni dei dispositivi client (API)

Gestisci le associazioni dei dispositivi client (console)

È possibile utilizzare la AWS IoT Greengrass console per visualizzare, aggiungere ed eliminare le associazioni dei dispositivi client.

Per visualizzare le associazioni dei dispositivi client per un dispositivo principale (console)

1. Passare alla console AWS IoT Greengrass.
2. Scegli Dispositivi principali.
3. Scegli il dispositivo principale da gestire.
4. Nella pagina dei dettagli del dispositivo principale, scegli la scheda Dispositivi client.
5. Nella sezione Dispositivi client associati, puoi vedere quali dispositivi client (AWS IoToggetti) sono associati al dispositivo principale.

Per associare i dispositivi client a un dispositivo principale (console)

1. Passare alla console AWS IoT Greengrass.
2. Scegli Dispositivi principali.
3. Scegli il dispositivo principale da gestire.
4. Nella pagina dei dettagli del dispositivo principale, scegli la scheda Dispositivi client.
5. Nella sezione Dispositivi client associati, scegli Associa dispositivi client.
6. Nella modalità Associa i dispositivi client al dispositivo principale, procedi come segue per ogni dispositivo client da associare:
   a. Immettete il nome dell'AWS IoToggetto da associare come dispositivo client.
   b. Scegli Aggiungi.
7. Selezionare Associate (Associa).

   I dispositivi client che hai associato possono ora utilizzare l'API di scoperta Greengrass per scoprire questo dispositivo principale.

Per dissociare i dispositivi client da un dispositivo principale (console)

1. Passare alla console AWS IoT Greengrass.
2. Scegli Dispositivi principali.
3. Scegli il dispositivo principale da gestire.

4. Nella pagina dei dettagli del dispositivo principale, scegli la scheda Dispositivi client.
5. Nella sezione Dispositivi client associati, seleziona ogni dispositivo client da dissociare.
6. Scegli Dissocia.
7. Nella modalità di conferma, scegli Dissocia.

   I dispositivi client che hai dissociato non possono più utilizzare l'API di scoperta Greengrass per scoprire questo dispositivo principale.

Gestisci le associazioni dei dispositivi client () AWS CLI

È possibile utilizzare AWS Command Line Interface (AWS CLI) per gestire le associazioni dei dispositivi client per un dispositivo principale.

Per visualizzare le associazioni dei dispositivi client per un dispositivo principale (AWS CLI)

- Utilizzate il seguente comando: list-client-devices-associated- with-core-device.

Per associare i dispositivi client a un dispositivo principale (AWS CLI)

- Utilizzate il seguente comando: batch-associate-client-device- with-core-device.

Per dissociare i dispositivi client da un dispositivo principale () AWS CLI

- Utilizzate il seguente comando: batch-disassociate-client-device- from-core-device.

Gestisci le associazioni dei dispositivi client (API)

È possibile utilizzare l'AWSAPI per gestire le associazioni dei dispositivi client per un dispositivo principale.

Per visualizzare le associazioni dei dispositivi client per un dispositivo principale (AWSAPI)

- Utilizzare la seguente operazione: ListClientDevicesAssociatedWithCoreDevice.

Per associare i dispositivi client a un dispositivo principale (AWSAPI)

- Utilizzare la seguente operazione: BatchAssociateClientDeviceWithCoreDevice.

Per dissociare i dispositivi client da un dispositivo principale (AWSAPI)

- Utilizzare la seguente operazione: BatchDisassociateClientDeviceFromCoreDevice.

Autenticazione dei client in modalità offline

Con l'autenticazione offline puoi configurare il tuo dispositivo AWS IoT Greengrass Core in modo che i dispositivi client possano connettersi a un dispositivo principale, anche quando il dispositivo principale non è connesso al cloud. Quando utilizzi l'autenticazione offline, i tuoi dispositivi Greengrass possono continuare a funzionare in un ambiente parzialmente offline.

Per utilizzare l'autenticazione offline per un dispositivo client con una connessione al cloud, è necessario quanto segue:

- Un dispositivo AWS IoT Greengrass Core con il Autenticazione del dispositivo client componente distribuito. È necessario utilizzare la versione 2.3.0 o successiva per l'autenticazione offline.
- Una connessione cloud per il dispositivo principale durante la connessione iniziale dei dispositivi client.

Archiviazione delle credenziali del client

Quando un dispositivo client si connette a un dispositivo principale per la prima volta, il dispositivo principale chiama il AWS IoT Greengrass servizio. Quando viene chiamato, Greengrass convalida la registrazione del dispositivo client come una cosa sola. AWS IoT Verifica inoltre che il dispositivo disponga di un certificato valido. Il dispositivo principale memorizza quindi queste informazioni localmente.

Alla successiva connessione del dispositivo, il dispositivo principale Greengrass tenta di convalidare il dispositivo client con il servizio. AWS IoT Greengrass Se non riesce a connettersi AWS IoT Greengrass, il dispositivo principale utilizza le informazioni sul dispositivo memorizzate localmente per convalidare il dispositivo client.

È possibile configurare il periodo di tempo in cui il dispositivo principale Greengrass memorizza le credenziali. È possibile impostare il timeout da un minuto a 2.147.483.647 minuti impostando l'opzione di `clientDeviceTrustDurationMinutes` configurazione nella configurazione del componente di autenticazione del dispositivo client. L'impostazione predefinita è un minuto, che disattiva efficacemente l'autenticazione offline. Quando imposti questo timeout, ti consigliamo di

considerare le tue esigenze di sicurezza. Dovresti anche considerare per quanto tempo ti aspetti che i dispositivi principali funzionino quando sono disconnessi dal cloud.

Il dispositivo principale aggiorna l'archiviazione delle credenziali tre volte:

1. Quando un dispositivo si connette al dispositivo principale per la prima volta.
2. Se il dispositivo principale è connesso al cloud, quando un dispositivo client si riconnette al dispositivo principale.
3. Se il dispositivo principale è connesso al cloud, una volta al giorno per aggiornare l'intero archivio di credenziali.

Quando il dispositivo principale Greengrass aggiorna l'archivio delle credenziali, utilizza l'operazione. ListClientDevicesAssociatedWithCoreDevice Greengrass aggiorna solo i dispositivi restituiti da questa operazione. Per associare un dispositivo client a un dispositivo principale, vedere. Associa i dispositivi client

Per utilizzare l'`ListClientDevicesAssociatedWithCoreDevice`operazione, è necessario aggiungere l'autorizzazione per l'operazione al ruolo AWS Identity and Access Management (IAM) associato a Account AWS quella in esecuzione AWS IoT Greengrass. Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

Gestisci gli endpoint principali dei dispositivi

Quando si utilizza il cloud discovery, si archiviano gli endpoint del broker MQTT per i dispositivi principali nel servizio cloud. AWS IoT Greengrass I dispositivi client si connettono AWS IoT Greengrass per recuperare questi endpoint e altre informazioni per i dispositivi principali associati.

Per ogni dispositivo principale, puoi gestire automaticamente o manualmente gli endpoint.

- Gestisci automaticamente gli endpoint con il rilevatore IP

  Puoi implementare il componente IP Detector per gestire automaticamente gli endpoint dei dispositivi principali al posto tuo se disponi di una configurazione di rete non complessa, ad esempio dove i dispositivi client si trovano sulla stessa rete del dispositivo principale. Non è possibile utilizzare il componente del rilevatore IP se il dispositivo principale si trova dietro un router che inoltra la porta del broker MQTT al dispositivo principale, ad esempio.

Il componente IP detector è utile anche se si esegue la distribuzione su gruppi di oggetti, poiché gestisce gli endpoint per tutti i dispositivi principali del gruppo di oggetti. Per ulteriori informazioni, consulta Usa il rilevatore IP per gestire automaticamente gli endpoint.

- Gestisci manualmente gli endpoint

  Se non puoi utilizzare il componente del rilevatore IP, devi gestire manualmente gli endpoint principali dei dispositivi. Puoi aggiornare questi endpoint con la console o l'API. Per ulteriori informazioni, consulta Gestisci manualmente gli endpoint.

Argomenti

Usa il rilevatore IP per gestire automaticamente gli endpoint

Se disponi di una configurazione di rete semplice, ad esempio i dispositivi client sulla stessa rete del dispositivo principale, puoi implementare il componente del rilevatore IP per eseguire le seguenti operazioni:

- Monitora le informazioni sulla connettività di rete locale del dispositivo Greengrass core. Queste informazioni includono gli endpoint di rete del dispositivo principale e la porta su cui opera il broker MQTT.
- Segnala le informazioni sulla connettività del dispositivo principale al AWS IoT Greengrass servizio cloud.

Il componente del rilevatore IP sovrascrive gli endpoint impostati manualmente.

> **Important**
>
> La AWS IoT politica del dispositivo principale deve consentire l'`greengrass:UpdateConnectivityInfo`autorizzazione all'uso del componente del rilevatore IP. Per ulteriori informazioni, consultare Policy AWS IoT per operazioni del piano dei dati e Configura la policy relativa agli AWS IoT oggetti.

È possibile effettuare una delle seguenti operazioni per distribuire il componente del rilevatore IP:

- Utilizza la pagina Configure Discovery nella console. Per ulteriori informazioni, consulta Configura cloud discovery (console).
- Crea e modifica le distribuzioni per includere il rilevatore IP. Puoi utilizzare la console o l'AWSAPI per gestire AWS CLI le distribuzioni. Per ulteriori informazioni, consulta Creare distribuzione.

Implementa il componente del rilevatore IP (console)

1. Nel menu di navigazione della AWS IoT Greengrassconsole, scegli Componenti.
2. Nella pagina Componenti, scegli la scheda Componenti pubblici, quindi scegli aws.greengrass.clientdevices.IPDetector.
3. Nella pagina aws.greengrass.clientdevices.IPDetector, scegli (Distribuisci).
4. Da Aggiungi alla distribuzione, scegli una distribuzione esistente da modificare oppure scegli di creare una nuova distribuzione, quindi scegli Avanti.
5. Se hai scelto di creare una nuova distribuzione, scegli il dispositivo principale o il gruppo di oggetti di destinazione per la distribuzione. Nella pagina Specificare la destinazione, in Obiettivo di distribuzione, scegli un dispositivo principale o un gruppo di oggetti, quindi scegli Avanti.
6. Nella pagina Seleziona componenti, verifica che il aws.greengrass.clientdevices.IPDetectorcomponente sia selezionato, scegli Avanti.
7. Nella pagina Configura componenti aws.greengrass.clientdevices.IPDetector, selezionate e quindi effettuate le seguenti operazioni:

   a. Scegli Configura componente.

   b. Nella aws.greengrass.clientdevices.IPDetector modalità Configura, in Aggiornamento della configurazione, in Configurazione da unire, è possibile inserire un aggiornamento della configurazione per configurare il componente del rilevatore IP. È possibile specificare una delle seguenti opzioni di configurazione:

      - `defaultPort`— (Facoltativo) La porta del broker MQTT da segnalare quando questo componente rileva gli indirizzi IP. È necessario specificare questo parametro se si configura il broker MQTT per utilizzare una porta diversa dalla porta predefinita 8883.
      - `includeIPv4LoopbackAddrs`— (Facoltativo) È possibile abilitare questa opzione per rilevare e segnalare gli indirizzi di loopback IPv4. Si tratta di indirizzi IP, ad esempio quelli con cui un dispositivo può comunicare con se stesso`localhost`. Utilizzate questa opzione in ambienti di test in cui il dispositivo principale e il dispositivo client funzionano sullo stesso sistema.

- `includeIPv4LinkLocalAddrs`— (Facoltativo) È possibile abilitare questa opzione per rilevare e segnalare gli indirizzi locali del collegamento IPv4. Utilizzate questa opzione se la rete del dispositivo principale non dispone di indirizzi IP assegnati staticamente o del Dynamic Host Configuration Protocol (DHCP).

L'aggiornamento della configurazione potrebbe essere simile all'esempio seguente.

```
{
  "defaultPort": "8883",
  "includeIPv4LoopbackAddrs": false,
  "includeIPv4LinkLocalAddrs": false
}
```

c. Scegli Conferma per chiudere la modalità, quindi scegli Avanti.

8. Nella pagina Configura impostazioni avanzate, mantieni le impostazioni di configurazione predefinite e scegli Avanti.
9. Nella pagina Review (Verifica), scegli Deploy (Distribuisci).

   Il completamento della distribuzione può richiedere fino a un minuto.

Implementa il componente del rilevatore IP () AWS CLI

Per distribuire il componente del rilevatore IP, create un documento di distribuzione che `aws.greengrass.clientdevices.IPDetector` includa l'`components`oggetto e specificate l'aggiornamento della configurazione per il componente. Segui le istruzioni riportate in Creare distribuzione basso per creare una nuova distribuzione o modificare una distribuzione esistente.

È possibile specificare una delle seguenti opzioni per configurare il componente del rilevatore IP quando si crea il documento di distribuzione:

- `defaultPort`— (Facoltativo) La porta del broker MQTT da segnalare quando questo componente rileva gli indirizzi IP. È necessario specificare questo parametro se si configura il broker MQTT per utilizzare una porta diversa dalla porta predefinita 8883.
- `includeIPv4LoopbackAddrs`— (Facoltativo) È possibile abilitare questa opzione per rilevare e segnalare gli indirizzi di loopback IPv4. Si tratta di indirizzi IP, ad esempio quelli con cui un dispositivo può comunicare con se stesso`localhost`. Utilizzate questa opzione in ambienti di test in cui il dispositivo principale e il dispositivo client funzionano sullo stesso sistema.

- `includeIPv4LinkLocalAddrs`— (Facoltativo) È possibile abilitare questa opzione per rilevare e segnalare gli indirizzi locali del collegamento IPv4. Utilizzate questa opzione se la rete del dispositivo principale non dispone di indirizzi IP assegnati staticamente o del Dynamic Host Configuration Protocol (DHCP).

Il seguente esempio di documento di distribuzione parziale specifica di segnalare la porta 8883 come porta del broker MQTT.

```
{
  ...,
  "components": {
    ...,
    "aws.greengrass.clientdevices.IPDetector": {
      "componentVersion": "2.1.1",
      "configurationUpdate": {
        "merge": "{\"defaultPort\":\"8883\",}"
      }
    }
  }
}
```

Gestisci manualmente gli endpoint

È possibile gestire manualmente gli endpoint del broker MQTT per i dispositivi principali.

Ogni endpoint del broker MQTT contiene le seguenti informazioni:

Punto finale () `HostAddress`

Un indirizzo IP o DNS in cui i dispositivi client possono connettersi a un broker MQTT sul dispositivo principale.

Porta (`PortNumber`)

La porta in cui opera il broker MQTT sul dispositivo principale.

È possibile configurare questa porta sul componente del broker Moquette MQTT, che per impostazione predefinita utilizza la porta 8883.

Metadati () `Metadata`

Metadati aggiuntivi da fornire ai dispositivi client che si connettono a questo endpoint.

Argomenti

Gestisci gli endpoint (console)

Puoi utilizzare la AWS IoT Greengrass console per visualizzare, aggiornare e rimuovere gli endpoint per un dispositivo principale.

Per gestire gli endpoint per un dispositivo principale (console)

1. Passare alla console AWS IoT Greengrass.
2. Scegli Dispositivi principali.
3. Scegli il dispositivo principale da gestire.
4. Nella pagina dei dettagli del dispositivo principale, scegli la scheda Dispositivi client.
5. Nella sezione Endpoint del broker MQTT, puoi vedere gli endpoint del broker MQTT del dispositivo principale. Scegli Gestisci gli endpoint.
6. Nella modalità Gestisci gli endpoint, aggiungi o rimuovi gli endpoint del broker MQTT per il dispositivo principale.
7. Scegli Aggiorna.

Gestisci gli endpoint () AWS CLI

Puoi usare il AWS Command Line Interface (AWS CLI) per gestire gli endpoint per un dispositivo principale.

> **Note**
>
> Poiché il supporto per i dispositivi client in AWS IoT Greengrass V2 è retrocompatibile conAWS IoT Greengrass V1, puoi utilizzare le nostre operazioni AWS IoT Greengrass V1 API AWS IoT Greengrass V2 per gestire gli endpoint principali dei dispositivi.

Per ottenere endpoint per un dispositivo principale () AWS CLI

- Utilizzate uno dei seguenti comandi:

- greengrass v2: get-connectivity-info
- erba verde: get-connectivity-info

Per aggiornare gli endpoint per un dispositivo principale () AWS CLI

- Utilizzate uno dei seguenti comandi:
  - greengrass v2: update-connectivity-info
  - erba verde: update-connectivity-info

Gestisci gli endpoint (API)

Puoi utilizzare l'AWSAPI per gestire gli endpoint per un dispositivo principale.

> **Note**
>
> Poiché il supporto per i dispositivi client in AWS IoT Greengrass V2 è retrocompatibile conAWS IoT Greengrass V1, puoi utilizzare le nostre operazioni AWS IoT Greengrass V1 API AWS IoT Greengrass V2 per gestire gli endpoint principali dei dispositivi.

Per ottenere endpoint per un dispositivo principale (API) AWS

- Utilizzate una delle seguenti operazioni:
  - V2: GetConnectivityInfo
  - V1: GetConnectivityInfo

Per aggiornare gli endpoint per un dispositivo principale (API) AWS

- Utilizzate una delle seguenti operazioni:
  - V2: UpdateConnectivityInfo
  - V1: UpdateConnectivityInfo

Scegli un broker MQTT

AWS IoT Greengrassoffre opzioni per scegliere quale broker MQTT locale eseguire sui dispositivi principali. I dispositivi client si connettono al broker MQTT che funziona su un dispositivo principale, quindi scegli un broker MQTT compatibile con i dispositivi client che desideri connettere.

> **Note**
>
> Ti consigliamo di implementare solo un componente del broker MQTT. I componenti del bridge MQTT e del rilevatore IP funzionano con un solo componente del broker MQTT alla volta. Se si distribuiscono più componenti del broker MQTT, è necessario configurarli per utilizzare porte diverse.

È possibile scegliere tra i seguenti broker MQTT:

- Broker MQTT 3.1.1 (Moquette) — `aws.greengrass.clientdevices.mqtt.Moquette`

  Scegliete questa opzione per un broker MQTT leggero e conforme allo standard MQTT 3.1.1. Il broker AWS IoT Core MQTT e io SDK per dispositivi AWS IoT siamo inoltre conformi allo standard MQTT 3.1.1, quindi puoi utilizzare queste funzionalità per creare un'applicazione che utilizzi MQTT 3.1.1 su tutti i tuoi dispositivi e il. Cloud AWS
- Broker MQTT 5 (EMQX) — `aws.greengrass.clientdevices.mqtt.EMQX`

  Scegliete questa opzione per utilizzare le funzionalità MQTT 5 nella comunicazione tra dispositivi principali e dispositivi client. Questo componente utilizza più risorse rispetto al broker Moquette MQTT 3.1.1 e sui dispositivi core Linux richiede Docker.

  MQTT 5 è retrocompatibile con MQTT 3.1.1, quindi è possibile connettere i dispositivi client che utilizzano MQTT 3.1.1 a questo broker. Se si utilizza il broker Moquette MQTT 3.1.1, è possibile sostituirlo con il broker EMQX MQTT 5 e i dispositivi client possono continuare a connettersi e funzionare normalmente.

- Implementa un broker personalizzato

  Scegli questa opzione per creare un componente broker locale personalizzato per comunicare con i dispositivi client. È possibile creare un broker locale personalizzato che utilizza un protocollo diverso da MQTT. AWS IoT Greengrassfornisce un componente SDK che è possibile utilizzare per autenticare e autorizzare i dispositivi client. Per ulteriori informazioni, consultare Usa il SDK per

dispositivi AWS IoT per comunicare con il nucleo Greengrass, altri componenti e AWS IoT Core e Autentica e autorizza i dispositivi client.

Connessione dei dispositivi client a un dispositivoAWS IoT Greengrass Core con un broker MQTT

Quando si utilizza un broker MQTT sul dispositivoAWS IoT Greengrass Core, il dispositivo utilizza un'autorità di certificazione (CA) del dispositivo principale esclusiva del dispositivo per rilasciare un certificato al broker per effettuare connessioni TLS reciproche con i client.

AWS IoT Greengrassgenererà automaticamente una CA del proprio. La CA del dispositivo principale viene registrataAWS IoT Greengrass quando ilAutenticazione del dispositivo client componente è collegato. La CA del dispositivo principale generata automaticamente è persistente, il dispositivo continuerà a utilizzare la stessa CA fintanto che il componente di autenticazione del dispositivo client è configurato.

Quando il broker MQTT si avvia, richiede un certificato. Il componente di autenticazione del dispositivo client rilascia un certificato X.509 utilizzando la CA del dispositivo principale. Il certificato viene ruotato all'avvio del broker, alla scadenza del certificato o quando le informazioni di connettività come l'indirizzo IP cambiano. Per ulteriori informazioni, consulta Rotazione dei certificati sul broker MQTT locale.

Per connettere un client al broker MQTT, è necessario:

- Il dispositivo client deve disporre della CA del dispositivoAWS IoT Greengrass Core. Puoi ottenere questa CA tramite cloud discovery o fornendola manualmente. Per ulteriori informazioni, consulta Utilizzo della propria autorità di certificazione.
- Il nome di dominio completo (FQDN) o l'indirizzo IP del dispositivo principale deve essere presente nel certificato del broker emesso dal CA del dispositivo. Puoi assicurarlo utilizzando ilRilevatore IP componente o configurando manualmente l'indirizzo IP. Per ulteriori informazioni, consulta Gestisci gli endpoint principali dei dispositivi.
- Il componente di autenticazione del dispositivo client deve consentire al dispositivo client di connettersi al dispositivo principale di Greengrass. Per ulteriori informazioni, consulta Autenticazione del dispositivo client.

Utilizzo della propria autorità di certificazione

Se i tuoi dispositivi client non riescono ad accedere al cloud per scoprire il tuo dispositivo principale, puoi fornire un'autorità di certificazione (CA) del dispositivo principale. Il tuo dispositivo principale Greengrass utilizza il dispositivo principale CA per emettere certificati per il tuo broker MQTT. Dopo aver configurato il dispositivo principale e aver fornito al dispositivo client la relativa CA, i dispositivi client possono connettersi all'endpoint e verificare l'handshake TLS utilizzando la CA del dispositivo principale (CA fornita in proprio o generata automaticamente).

Per configurare ilAutenticazione del dispositivo client componente in modo che utilizzi la CA del dispositivo principale, impostate il parametro di`certificateAuthority` configurazione al momento della distribuzione del componente. È necessario fornire i seguenti:

- La posizione di un certificato CA del dispositivo principale.
- La chiave privata del certificato CA del dispositivo principale.
- (Facoltativo) La catena di certificati al certificato principale se la CA del dispositivo principale è una CA intermedia.

Se fornisci una CA del dispositivo principale,AWS IoT Greengrass registra la CA nel cloud.

È possibile archiviare i certificati in un modulo di sicurezza hardware o nel file system. L'esempio seguente mostra una`certificateAuthority` configurazione per una CA intermedia archiviata utilizzando HSM/TPM. Nota che la catena di certificati può essere archiviata solo su disco.

```
"certificateAuthority": {
    "certificateUri": "pkcs11:object=CustomerIntermediateCA;type=cert",
    "privateKeyUri": "pkcs11:object=CustomerIntermediateCA;type=private"
    "certificateChainUri": "file:///home/ec2-user/creds/certificateChain.pem",
  }
```

In questo esempio, il parametro di`certificateAuthority` configurazione configura il componente di autenticazione del dispositivo client per utilizzare una CA intermedia del file system:

```
"certificateAuthority": {
    "certificateUri": "file:///home/ec2-user/creds/intermediateCA.pem",
    "privateKeyUri": "file:///home/ec2-user/creds/intermediateCA.privateKey.pem",
    "certificateChainUri": "file:///home/ec2-user/creds/certificateChain.pem",
  }
```

Per connettere i dispositivi al tuo dispositivoAWS IoT Greengrass Core, procedi come segue:

1. Crea un'autorità di certificazione (CA) intermedia per il dispositivo principale di Greengrass utilizzando la CA root della propria. Si consiglia di utilizzare una CA intermedia come best practice di sicurezza.
2. Fornisci il certificato CA intermedio, la chiave privata e la catena di certificati alla CA principale al dispositivo principale di Greengrass. Per ulteriori informazioni, consulta Autenticazione del dispositivo client. La CA intermedia diventa la CA del dispositivo principale per il dispositivo core Greengrass e il dispositivo registra la CA conAWS IoT Greengrass.
3. Registra il dispositivo client come qualsiasiAWS IoT cosa. Per ulteriori informazioni, consulta Create a thing object nella Guida per gliAWS IoT Core sviluppatori. Aggiungi la chiave privata, la chiave pubblica, il certificato del dispositivo e il certificato CA root al dispositivo client. La modalità di aggiunta delle informazioni dipende dal dispositivo e dal software in uso.

Una volta configurato il dispositivo, puoi utilizzare il certificato e la catena di chiavi pubbliche per connetterti al dispositivo principale di Greengrass. Il software è responsabile dell'individuazione degli endpoint principali del dispositivo. È possibile impostare manualmente l'endpoint per il dispositivo principale. Per ulteriori informazioni, consulta Gestisci manualmente gli endpoint.

Verifica le comunicazioni con i dispositivi client

I dispositivi client possono utilizzare il SDK per dispositivi AWS IoT per rilevare, connettersi e comunicare con un dispositivo principale. È possibile utilizzare il Greengrass discovery client SDK per dispositivi AWS IoT per utilizzare l'API Greengrass discovery, che restituisce informazioni sui dispositivi principali a cui un dispositivo client può connettersi. La risposta dell'API include gli endpoint del broker MQTT da connettere e i certificati da utilizzare per verificare l'identità di ciascun dispositivo principale. Quindi, il dispositivo client può provare ogni endpoint fino a quando non si connette correttamente a un dispositivo principale.

I dispositivi client possono rilevare solo i dispositivi principali a cui vengono associati. Prima di testare le comunicazioni tra un dispositivo client e un dispositivo principale, è necessario associare il dispositivo client al dispositivo principale. Per ulteriori informazioni, consulta Associa i dispositivi client.

L'API Greengrass discovery restituisce gli endpoint del broker MQTT del dispositivo principale specificati. Puoi utilizzare il componente IP Detector per gestire questi endpoint al posto tuo oppure puoi gestirli manualmente per ogni dispositivo principale. Per ulteriori informazioni, consulta Gestisci gli endpoint principali dei dispositivi.

> **Note**
>
> Per utilizzare l'API di scoperta Greengrass, un dispositivo client deve disporre dell'`greengrass:Discover`autorizzazione. Per ulteriori informazioni, consulta AWS IoTPolicy minima per i dispositivi client.

SDK per dispositivi AWS IoTÈ disponibile in più linguaggi di programmazione. Per ulteriori informazioni, consulta AWS IoTDevice SDK nella AWS IoT CoreDevice Guide.

Argomenti

- Comunicazioni di test (Python)
- Comunicazioni di test (C++)
- Comunicazioni di prova () JavaScript
- Comunicazioni di test (Java)

Comunicazioni di test (Python)

In questa sezione, si utilizza l'esempio di scoperta di Greengrass nella versione 2 SDK per dispositivi AWS IoTper Python per testare le comunicazioni tra un dispositivo client e un dispositivo principale.

> **Important**
>
> Per utilizzare la versione SDK per dispositivi AWS IoT v2 per Python, un dispositivo deve eseguire Python 3.6 o versione successiva.

Per testare le comunicazioni (SDK per dispositivi AWS IoTv2 per Python)

1. Scarica e installa la SDK per dispositivi AWS IoTversione 2 per Python AWS IoT sull'oggetto da connettere come dispositivo client.

   Sul dispositivo client, procedi come segue:

   a. Clona il repository SDK per dispositivi AWS IoT v2 for Python per scaricarlo.

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
   ```

b. Installa la SDK per dispositivi AWS IoT v2 per Python.

```
python3 -m pip install --user ./aws-iot-device-sdk-python-v2
```

2. Passa alla cartella samples nella SDK per dispositivi AWS IoT v2 per Python.

```
cd aws-iot-device-sdk-python-v2/samples
```

3. Esegui l'applicazione Greengrass discovery di esempio. Questa applicazione prevede argomenti che specifichino il nome dell'oggetto del dispositivo client, l'argomento e il messaggio MQTT da utilizzare e i certificati che autenticano e proteggono la connessione. L'esempio seguente invia un messaggio Hello World all'argomento. `clients/`*`MyClientDevice1`*`/hello/world`

   - Sostituisci *`MyClientDevice1`* con il nome dell'oggetto del dispositivo client.
   - Sostituisci *`~/certs/ AmazonRoot CA1.pem con`* il percorso del certificato CA root Amazon sul dispositivo client.
   - Sostituisci *`~/certs/device.pem.crt con il percorso del certificato del dispositivo`* sul dispositivo client.
   - Sostituisci *`~/certs/private.pem.key`* con il percorso del file della chiave privata sul dispositivo client.
   - Sostituisci *`us-east-1`* con la regione in cui operano il dispositivo client e il dispositivo principale. AWS

```
python3 basic_discovery.py \\
  --thing_name MyClientDevice1 \\
  --topic 'clients/MyClientDevice1/hello/world' \\
  --message 'Hello World!' \\
  --ca_file ~/certs/AmazonRootCA1.pem \\
  --cert ~/certs/device.pem.crt \\
  --key ~/certs/private.pem.key \\
  --region us-east-1 \\
  --verbosity Warn
```

   L'applicazione di esempio Discovery invia il messaggio 10 volte e si disconnette. Inoltre, sottoscrive lo stesso argomento in cui pubblica i messaggi. Se l'output indica che l'applicazione ha ricevuto messaggi MQTT sull'argomento, il dispositivo client può comunicare correttamente con il dispositivo principale.

```
Performing greengrass discovery...
awsiot.greengrass_discovery.DiscoverResponse(gg_groups=[awsiot.greengrass_discovery.GGGroup
coreDevice-MyGreengrassCore',
 cores=[awsiot.greengrass_discovery.GGCore(thing_arn='arn:aws:iot:us-
east-1:123456789012:thing/MyGreengrassCore',
 connectivity=[awsiot.greengrass_discovery.ConnectivityInfo(id='203.0.113.0',
 host_address='203.0.113.0', metadata='', port=8883)])],
 certificate_authorities=['-----BEGIN CERTIFICATE-----\
MIICiT...EXAMPLE=\
-----END CERTIFICATE-----\
'])])
Trying core arn:aws:iot:us-east-1:123456789012:thing/MyGreengrassCore at host
 203.0.113.0 port 8883
Connected!
Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 0}

Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 0}'
Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 1}

Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 1}'

...

Published topic clients/MyClientDevice1/hello/world: {"message": "Hello World!",
 "sequence": 9}

Publish received on topic clients/MyClientDevice1/hello/world
b'{"message": "Hello World!", "sequence": 9}'
```

Se invece l'applicazione genera un errore, consulta Risoluzione dei problemi di rilevamento di Greengrass.

Puoi anche visualizzare i log di Greengrass sul dispositivo principale per verificare se il dispositivo client si connette e invia messaggi correttamente. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Comunicazioni di test (C++)

In questa sezione, si utilizza l'esempio di scoperta di Greengrass nella versione SDK per dispositivi AWS IoT2 per C++ per testare le comunicazioni tra un dispositivo client e un dispositivo principale.

Per creare la SDK per dispositivi AWS IoT v2 per C++, un dispositivo deve disporre dei seguenti strumenti:

- C++ 11 o versione successiva
- CMake 3.1 o successivo
- Uno dei seguenti compilatori:
  - GCC 4.8 o successivo
  - Clang 3.9 o successivo
  - MSVC 2015 o versione successiva

Per testare le comunicazioni (SDK per dispositivi AWS IoTv2 per C++)

1. Scarica e crea la versione SDK per dispositivi AWS IoTv2 per C++ su cui AWS IoT connetterti come dispositivo client.

   Sul dispositivo client, procedi come segue:

   a. Create una cartella per l'area di lavoro SDK per dispositivi AWS IoT v2 for C++ e modificatela.

   ```
   cd
   mkdir iot-device-sdk-cpp
   cd iot-device-sdk-cpp
   ```

   b. Clona il repository SDK per dispositivi AWS IoT v2 for C++ per scaricarlo. Il `--recursive` flag specifica di scaricare i sottomoduli.

   ```
   git clone --recursive https://github.com/aws/aws-iot-device-sdk-cpp-v2.git
   ```

   c. Crea una cartella per l'output della build SDK per dispositivi AWS IoT v2 for C++ e modificala.

   ```
   mkdir aws-iot-device-sdk-cpp-v2-build
   ```

```
cd aws-iot-device-sdk-cpp-v2-build
```

d. Crea la SDK per dispositivi AWS IoT v2 per C++.

```
cmake -DCMAKE_INSTALL_PREFIX="~/iot-device-sdk-cpp" -
DCMAKE_BUILD_TYPE="Release" ../aws-iot-device-sdk-cpp-v2
cmake --build . --target install
```

2. Crea l'applicazione di esempio Greengrass discovery nella versione SDK per dispositivi AWS IoT 2 per C++. Esegui questa operazione:

a. Passate alla cartella di esempio Greengrass discovery nella versione SDK per dispositivi AWS IoT 2 per C++.

```
cd ../aws-iot-device-sdk-cpp-v2/samples/greengrass/basic_discovery
```

b. Crea una cartella per l'output della build di esempio di Greengrass discovery e modificala.

```
mkdir build
cd build
```

c. Crea l'applicazione di esempio Greengrass discovery.

```
cmake -DCMAKE_PREFIX_PATH="~/iot-device-sdk-cpp" -
DCMAKE_BUILD_TYPE="Release" ..
cmake --build . --config "Release"
```

3. Esegui l'applicazione Greengrass discovery di esempio. Questa applicazione prevede argomenti che specifichino il nome dell'oggetto del dispositivo client, l'argomento MQTT da utilizzare e i certificati che autenticano e proteggono la connessione. L'esempio seguente sottoscrive l'`clients/`*`MyClientDevice1`*`/hello/world`argomento e pubblica un messaggio immesso nella riga di comando relativo allo stesso argomento.

- Sostituisci *`MyClientDevice1`* con il nome dell'oggetto del dispositivo client.
- Sostituisci *`~/certs/ AmazonRoot CA1.pem con`* il percorso del certificato CA root Amazon sul dispositivo client.
- Sostituisci *`~/certs/device.pem.crt con il percorso del certificato del dispositivo`* sul dispositivo client.
- Sostituisci *`~/certs/private.pem.key`* con il percorso del file della chiave privata sul dispositivo client.

- Sostituisci *us-east-1* con la regione in cui operano il dispositivo client e il dispositivo principale. AWS

```
./basic-discovery \
  --thing_name MyClientDevice1 \
  --topic 'clients/MyClientDevice1/hello/world' \
  --ca_file ~/certs/AmazonRootCA1.pem \
  --cert ~/certs/device.pem.crt \
  --key ~/certs/private.pem.key \
  --region us-east-1
```

L'applicazione di esempio discovery sottoscrive l'argomento e richiede all'utente di inserire un messaggio da pubblicare.

```
Connecting to group greengrassV2-coreDevice-MyGreengrassCore with thing arn
 arn:aws:iot:us-east-1:123456789012:thing/MyGreengrassCore, using endpoint
 203.0.113.0:8883
Connected to group greengrassV2-coreDevice-MyGreengrassCore, using connection to
 203.0.113.0:8883
Successfully subscribed to clients/MyClientDevice1/hello/world
Enter the message you want to publish to topic clients/MyClientDevice1/hello/world
 and press enter. Enter 'exit' to exit this program.
```

Se invece l'applicazione genera un errore, consulta Risoluzione dei problemi di rilevamento di Greengrass.

4. Inserisci un messaggio, ad esempio. **`Hello World!`**

```
Enter the message you want to publish to topic clients/MyClientDevice1/hello/world
 and press enter. Enter 'exit' to exit this program.
Hello World!
```

Se l'output indica che l'applicazione ha ricevuto il messaggio MQTT sull'argomento, il dispositivo client può comunicare correttamente con il dispositivo principale.

```
Operation on packetId 2 Succeeded
Publish received on topic clients/MyClientDevice1/hello/world
Message:
Hello World!
```

Puoi anche visualizzare i log di Greengrass sul dispositivo principale per verificare se il dispositivo client si connette e invia messaggi correttamente. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Comunicazioni di prova () JavaScript

In questa sezione, si utilizza Greengrass discovery sample nella versione SDK per dispositivi AWS IoT2 per JavaScript testare le comunicazioni tra un dispositivo client e un dispositivo principale.

> **Important**
>
> Per utilizzare la versione SDK per dispositivi AWS IoT v2 for JavaScript, un dispositivo deve eseguire Node v10.0 o versione successiva.

Per testare le comunicazioni (v2 for) SDK per dispositivi AWS IoT JavaScript

1. Scarica e installa la SDK per dispositivi AWS IoTversione 2 sull'AWS IoToggetto JavaScript da connettere come dispositivo client.

   Sul dispositivo client, procedi come segue:

   a. Clona il JavaScript repository SDK per dispositivi AWS IoT v2 for per scaricarlo.

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-js-v2.git
   ```

   b. Installa la v2 perSDK per dispositivi AWS IoT. JavaScript

   ```
   cd aws-iot-device-sdk-js-v2
   npm install
   ```

2. Passa alla cartella di esempio Greengrass discovery nella versione 2 perSDK per dispositivi AWS IoT. JavaScript

   ```
   cd samples/node/basic_discovery
   ```

3. Installa l'applicazione di esempio Greengrass discovery.

   ```
   npm install
   ```

4. Esegui l'applicazione Greengrass discovery di esempio. Questa applicazione prevede argomenti che specifichino il nome dell'oggetto del dispositivo client, l'argomento e il messaggio MQTT da utilizzare e i certificati che autenticano e proteggono la connessione. L'esempio seguente invia un messaggio Hello World all'argomento. `clients/`*`MyClientDevice1`*`/hello/world`

   - Sostituisci *`MyClientDevice1`* con il nome dell'oggetto del dispositivo client.
   - Sostituisci *`~/certs/ AmazonRoot CA1.pem con`* il percorso del certificato CA root Amazon sul dispositivo client.
   - Sostituisci *`~/certs/device.pem.crt con il percorso del certificato del dispositivo`* sul dispositivo client.
   - Sostituisci *`~/certs/private.pem.key`* con il percorso del file della chiave privata sul dispositivo client.
   - Sostituisci *`us-east-1`* con la regione in cui operano il dispositivo client e il dispositivo principale. AWS

```
node dist/index.js \
  --thing_name MyClientDevice1 \
  --topic 'clients/MyClientDevice1/hello/world' \
  --message 'Hello World!' \
  --ca_file ~/certs/AmazonRootCA1.pem \
  --cert ~/certs/device.pem.crt \
  --key ~/certs/private.pem.key \
  --region us-east-1 \
  --verbose warn
```

L'applicazione di esempio Discovery invia il messaggio 10 volte e si disconnette. Inoltre, sottoscrive lo stesso argomento in cui pubblica i messaggi. Se l'output indica che l'applicazione ha ricevuto messaggi MQTT sull'argomento, il dispositivo client può comunicare correttamente con il dispositivo principale.

```
Discovery Response:
{"gg_groups":[{"gg_group_id":"greengrassV2-coreDevice-
MyGreengrassCore","cores":[{"thing_arn":"arn:aws:iot:us-
east-1:123456789012:thing/MyGreengrassCore","connectivity":
[{"id":"203.0.113.0","host_address":"203.0.113.0","port":8883,"metadata":""}]}],"certificat
["-----BEGIN CERTIFICATE-----\nMIICiT...EXAMPLE=\n-----END CERTIFICATE-----\n"]}]}
Trying
 endpoint={"id":"203.0.113.0","host_address":"203.0.113.0","port":8883,"metadata":""}
```

```
[WARN] [2021-06-12T00:46:45Z] [00007f90c0e8d700] [socket] - id=0x7f90b8018710
 fd=26: setsockopt() for NO_SIGNAL failed with errno 92. If you are having SIGPIPE
 signals thrown, you may want to install a signal trap in your application layer.
Connected to
 endpoint={"id":"203.0.113.0","host_address":"203.0.113.0","port":8883,"metadata":""}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":1}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":2}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":3}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":4}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":5}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":6}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":7}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":8}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":9}
Publish received. topic:"clients/MyClientDevice1/hello/world" dup:false qos:0
 retain:false
{"message":"Hello World!","sequence":10}
Complete!
```

Se invece l'applicazione genera un errore, consulta Risoluzione dei problemi di rilevamento di Greengrass.

Puoi anche visualizzare i log di Greengrass sul dispositivo principale per verificare se il dispositivo client si connette e invia messaggi correttamente. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

Comunicazioni di test (Java)

In questa sezione, si utilizza Greengrass discovery sample nella versione SDK per dispositivi AWS IoT2 per Java per testare le comunicazioni tra un dispositivo client e un dispositivo principale.

> **Important**
>
> Per creare la versione SDK per dispositivi AWS IoT v2 per Java, un dispositivo deve disporre dei seguenti strumenti:
>
> - Java 8 o versione successiva, con `JAVA_HOME` puntamento alla cartella Java.
> - Apache Maven

Per testare le comunicazioni (SDK per dispositivi AWS IoTv2 per Java)

1. Scaricate e create la SDK per dispositivi AWS IoTversione v2 per Java su un AWS IoT dispositivo da connettere come dispositivo client.

   Sul dispositivo client, effettuate le seguenti operazioni:

   a. Clona il repository SDK per dispositivi AWS IoT v2 for Java per scaricarlo.

   ```
   git clone https://github.com/aws/aws-iot-device-sdk-java-v2.git
   ```

   b. Passa alla cartella SDK per dispositivi AWS IoT v2 for Java.

   c. Crea la SDK per dispositivi AWS IoT v2 per Java.

   ```
   cd aws-iot-device-sdk-java-v2
   mvn versions:use-latest-versions -Dincludes="software.amazon.awssdk.crt*"
   mvn clean install
   ```

2. Esegui l'applicazione Greengrass discovery di esempio. Questa applicazione prevede argomenti che specifichino il nome dell'oggetto del dispositivo client, l'argomento MQTT da utilizzare e i certificati che autenticano e proteggono la connessione. L'esempio seguente sottoscrive l'`clients/`*`MyClientDevice1`*`/hello/world`argomento e pubblica un messaggio immesso nella riga di comando relativo allo stesso argomento.

   - Sostituisci entrambe le istanze di *`MyClientDevice1`* con il nome dell'oggetto del dispositivo client.

- Sostituisci *$HOME/certs/ AmazonRoot CA1.pem con* il percorso del certificato CA root Amazon sul dispositivo client.
- Sostituisci *$HOME/certs/device.pem.crt con il percorso del certificato del dispositivo* sul dispositivo client.
- Sostituisci *$HOME/certs/private.pem.key* con il percorso del file della chiave privata sul dispositivo client.
- Sostituisci *us-east-1* con il luogo in cui operano il dispositivo client e il dispositivo principale. Regione AWS

```
DISCOVERY_SAMPLE_ARGS="--thing_name MyClientDevice1 \
  --topic 'clients/MyClientDevice1/hello/world' \
  --ca_file $HOME/certs/AmazonRootCA1.pem \
  --cert $HOME/certs/device.pem.crt \
  --key $HOME/certs/private.pem.key \
  --region us-east-1"

mvn exec:java -pl samples/Greengrass \
  -Dexec.mainClass=greengrass.BasicDiscovery \
  -Dexec.args="$DISCOVERY_SAMPLE_ARGS"
```

L'applicazione di esempio discovery sottoscrive l'argomento e richiede all'utente di inserire un messaggio da pubblicare.

```
Connecting to group ID greengrassV2-coreDevice-MyGreengrassCore, with thing
 arn arn:aws:iot:us-east-1:123456789012:thing/MyGreengrassCore, using endpoint
 203.0.113.0:8883
Started a clean session
Enter the message you want to publish to topic clients/MyClientDevice1/hello/world
 and press Enter. Type 'exit' or 'quit' to exit this program:
```

Se invece l'applicazione genera un errore, consulta Risoluzione dei problemi di rilevamento di Greengrass.

3. Inserisci un messaggio, ad esempio. **`Hello World!`**

```
Enter the message you want to publish to topic clients/MyClientDevice1/hello/world
 and press Enter. Type 'exit' or 'quit' to exit this program:
Hello World!
```

Se l'output indica che l'applicazione ha ricevuto il messaggio MQTT sull'argomento, il dispositivo client può comunicare correttamente con il dispositivo principale.

```
Message received on topic clients/MyClientDevice1/hello/world: Hello World!
```

Puoi anche visualizzare i log di Greengrass sul dispositivo principale per verificare se il dispositivo client si connette e invia messaggi correttamente. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

API RESTful per la scoperta di Greengrass

AWS IoT Greengrassfornisce il funzionamento dell'`Discover`API che i dispositivi client possono utilizzare per identificare i dispositivi core Greengrass a cui possono connettersi. I dispositivi client utilizzano questa operazione sul piano dati per recuperare le informazioni necessarie per connettersi ai dispositivi principali Greengrass, dove vengono associati al funzionamento BatchAssociateClientDeviceWithCoreDevicedell'API. Quando un dispositivo client è online, può connettersi al servizio AWS IoT Greengrass cloud e utilizzare l'API Discovery per trovare:

- L'indirizzo IP e la porta per ogni dispositivo principale Greengrass associato.
- Il certificato CA del dispositivo principale, che i dispositivi client possono utilizzare per autenticare il dispositivo principale Greengrass.

> **Note**
>
> I dispositivi client possono anche utilizzare il discovery client SDK per dispositivi AWS IoT per scoprire le informazioni di connettività per i dispositivi core Greengrass. Il client di scoperta utilizza l'API di rilevamento. Per ulteriori informazioni, consulta gli argomenti seguenti:
>
> - Verifica le comunicazioni con i dispositivi client
> - API RESTful di Greengrass Discovery nella Guida per gli AWS IoT Greengrass Version 1 sviluppatori.

Per utilizzare questa operazione API, invia richieste HTTP all'API di rilevamento sull'endpoint del piano dati Greengrass. Questo endpoint API ha il seguente formato.

```
https://greengrass-ats.iot.region.amazonaws.com:port/greengrass/discover/thing/thing-name
```

Per un elenco degli endpoint Regioni AWS e degli endpoint supportati per l'API AWS IoT Greengrass Discovery, consulta AWS IoT Greengrass V2endpoint e quote in. Riferimenti generali di AWS Questa operazione API è disponibile solo sull'endpoint del piano dati Greengrass. L'endpoint del piano di controllo utilizzato per gestire i componenti e le distribuzioni è diverso dall'endpoint del piano dati.

> **Note**
>
> L'API discovery è la stessa per e. AWS IoT Greengrass V1 AWS IoT Greengrass V2 Se disponi di dispositivi client che si connettono a un AWS IoT Greengrass V1 core, puoi connetterli ai dispositivi AWS IoT Greengrass V2 principali senza modificare il codice sui dispositivi client. Per ulteriori informazioni, consulta l'API RESTful di Greengrass Discovery nella Developer Guide. AWS IoT Greengrass Version 1

Argomenti

Autenticazione e autorizzazione Discovery

Per utilizzare l'API di rilevamento per recuperare le informazioni di connettività, un dispositivo client deve utilizzare l'autenticazione reciproca TLS con un certificato client X.509 per l'autenticazione. Per ulteriori informazioni, consulta i certificati client X.509 nella Developer Guide. AWS IoT Core

Un dispositivo client deve inoltre disporre dell'autorizzazione per eseguire l'`greengrass:Discover`azione. La seguente AWS IoT politica di esempio consente `MyClientDevice1` a un AWS IoT oggetto denominato di funzionare `Discover` da solo.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Action": "greengrass:Discover",
      "Resource": [
        "arn:aws:iot:us-west-2:123456789012:thing/MyClientDevice1"
      ]
    }
  ]
}
```

> **Important**
>
> Le variabili Thing Policy (`iot:Connection.Thing.*`) non sono supportate nelle AWS IoT politiche per i dispositivi principali o nelle operazioni del piano dati Greengrass. Puoi invece utilizzare un carattere jolly che corrisponda a più dispositivi con nomi simili. Ad esempio, puoi specificare `MyGreengrassDevice*` di `MyGreengrassDevice1` abbinare e così via. `MyGreengrassDevice2`

Per ulteriori informazioni, consulta AWS IoT Corele politiche nella Guida per AWS IoT Core gli sviluppatori.

Richiesta

La richiesta contiene le intestazioni HTTP standard e viene inviata all'endpoint di scoperta Greengrass, come illustrato negli esempi seguenti.

Il numero di porta dipende dal fatto che il dispositivo principale sia configurato per inviare traffico HTTPS sulla porta 8443 o sulla porta 443. Per ulteriori informazioni, consulta the section called “Connessione alla porta 443 o tramite un proxy di rete”.

> **Note**
>
> Questi esempi utilizzano l'endpoint Amazon Trust Services (ATS), che funziona con i certificati CA root ATS consigliati. Gli endpoint devono corrispondere al tipo di certificato CA root.

Porta 8443

```
HTTP GET https://greengrass-ats.iot.region.amazonaws.com:8443/greengrass/discover/
thing/thing-name
```

Porta 443

```
HTTP GET https://greengrass-ats.iot.region.amazonaws.com:443/greengrass/discover/
thing/thing-name
```

> **Note**
>
> I client che si connettono alla porta 443 devono implementare l'estensione TLS Application Layer Protocol Negotiation (ALPN) e `x-amzn-http-ca` passarla come in. `ProtocolName` `ProtocolNameList` Per ulteriori informazioni, consulta Protocols nella Developer Guide. AWS IoT

Risposta

In caso di esito positivo, l'intestazione della risposta include il codice di stato HTTP 200 e il corpo della risposta contiene il documento di risposta discover.

> **Note**
>
> Poiché AWS IoT Greengrass V2 utilizza la stessa API di rilevamentoAWS IoT Greengrass V1, la risposta organizza le informazioni in base a AWS IoT Greengrass V1 concetti, ad esempio i gruppi Greengrass. La risposta contiene un elenco di gruppi Greengrass. InAWS IoT Greengrass V2, ogni dispositivo principale fa parte del proprio gruppo, dove il gruppo contiene solo quel dispositivo principale e le relative informazioni di connettività.

Documenti di risposta di individuazione di esempio

Il seguente documento mostra la risposta per un dispositivo client associato a un dispositivo principale Greengrass. Il dispositivo principale ha un endpoint e un certificato CA.

```
{
  "GGGroups": [
```

```
    {
      "GGGroupId": "greengrassV2-coreDevice-core-device-01-thing-name",
      "Cores": [
        {
          "thingArn": "core-device-01-thing-arn",
          "Connectivity": [
            {
              "id": "core-device-01-connection-id",
              "hostAddress": "core-device-01-address",
              "portNumber": core-device-01-port,
              "metadata": "core-device-01-description"
            }
          ]
        }
      ],
      "CAs": [
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----"
      ]
    }
  ]
}
```

Il documento seguente mostra la risposta per un dispositivo client associato a due dispositivi principali. I dispositivi principali hanno più endpoint e più certificati CA di gruppo.

```
{
  "GGGroups": [
    {
      "GGGroupId": "greengrassV2-coreDevice-core-device-01-thing-name",
      "Cores": [
        {
          "thingArn": "core-device-01-thing-arn",
          "Connectivity": [
            {
              "id": "core-device-01-connection-id",
              "hostAddress": "core-device-01-address",
              "portNumber": core-device-01-port,
              "metadata": "core-device-01-connection-1-description"
            },
            {
              "id": "core-device-01-connection-id-2",
              "hostAddress": "core-device-01-address-2",
              "portNumber": core-device-01-port-2,
```

```
              "metadata": "core-device-01-connection-2-description"
            }
          ]
        }
      ],
      "CAs": [
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----"
      ]
    },
    {
      "GGGroupId": "greengrassV2-coreDevice-core-device-02-thing-name",
      "Cores": [
        {
          "thingArn":"core-device-02-thing-arn",
          "Connectivity" : [
            {
              "id": "core-device-02-connection-id",
              "hostAddress": "core-device-02-address",
              "portNumber": core-device-02-port,
              "metadata": "core-device-02-connection-1-description"
            }
          ]
        }
      ],
      "CAs": [
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----",
        "-----BEGIN CERTIFICATE-----cert-contents-----END CERTIFICATE-----"
      ]
    }
  ]
}
```

Prova l'API di scoperta con cURL

Se l'hai cURL installata, puoi testare l'API di rilevamento. L'esempio seguente specifica i certificati di un dispositivo client per autenticare una richiesta all'endpoint dell'API di scoperta Greengrass.

```
curl -i \
  --cert 1a23bc4d56.cert.pem \
  --key 1a23bc4d56.private.key \
```

```
  https://greengrass-ats.iot.us-west-2.amazonaws.com:8443/greengrass/discover/
thing/MyClientDevice1
```

> **Note**
>
> L'`-i`argomento specifica l'output delle intestazioni di risposta HTTP. È possibile utilizzare questa opzione per identificare gli errori.

Se la richiesta ha esito positivo, questo comando restituisce una risposta simile all'esempio seguente.

```
{
  "GGGroups": [
    {
      "GGGroupId": "greengrassV2-coreDevice-MyGreengrassCore",
      "Cores": [
        {
          "thingArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
          "Connectivity": [
            {
              "Id": "AUTOIP_192.168.1.4_1",
              "HostAddress": "192.168.1.5",
              "PortNumber": 8883,
              "Metadata": ""
            }
          ]
        }
      ],
      "CAs": [
        "-----BEGIN CERTIFICATE-----\ncert-contents\n-----END CERTIFICATE-----\n"
      ]
    }
  ]
}
```

Se il comando genera un errore, vedi Risoluzione dei problemi di rilevamento di Greengrass.

Inoltra messaggi MQTT tra dispositivi client e AWS IoT Core

È possibile inoltrare messaggi MQTT e altri dati tra dispositivi client e. AWS IoT Core I dispositivi client si connettono al componente del broker MQTT che viene eseguito sul dispositivo principale.

Per impostazione predefinita, i dispositivi principali non inoltrano messaggi o dati MQTT tra dispositivi client e. AWS IoT Core Per impostazione predefinita, i dispositivi client possono comunicare tra loro solo tramite MQTT.

Per inoltrare messaggi MQTT tra dispositivi client eAWS IoT Core, configurate il componente bridge MQTT per effettuare le seguenti operazioni:

- Inoltra i messaggi dai dispositivi client a. AWS IoT Core
- Inoltra i messaggi dai AWS IoT Core dispositivi client.

> **Note**
>
> Il bridge MQTT utilizza QoS 1 per pubblicare e AWS IoT Core sottoscrivere, anche quando un dispositivo client utilizza QoS 0 per pubblicare e sottoscrivere il broker MQTT locale. Di conseguenza, è possibile osservare una latenza aggiuntiva quando si inoltrano messaggi MQTT dai dispositivi client sul broker MQTT locale a. AWS IoT Core Per ulteriori informazioni sulla configurazione MQTT sui dispositivi principali, vedere. Configurare i timeout MQTT e le impostazioni della cache

Argomenti

Configurare e distribuire il componente bridge MQTT

Il componente bridge MQTT utilizza un elenco di mappature di argomenti, ciascuna delle quali specifica un'origine e una destinazione del messaggio. Per inoltrare messaggi tra dispositivi client eAWS IoT Core, implementate il componente bridge MQTT e specificate ogni argomento di origine e destinazione nella configurazione del componente.

Per distribuire il componente bridge MQTT su un dispositivo principale o su un gruppo di dispositivi principali, create una distribuzione che includa il componente. `aws.greengrass.clientdevices.mqtt.Bridge` Specificate le mappature degli argomenti nella configurazione `mqttTopicMapping` del componente bridge MQTT nella distribuzione.

L'esempio seguente definisce una distribuzione che configura il componente bridge MQTT per inoltrare messaggi su argomenti che corrispondono al filtro degli argomenti dai dispositivi client `clients/+/hello/world` a. AWS IoT Core L'aggiornamento della `merge` configurazione richiede un oggetto JSON serializzato. Per ulteriori informazioni, consulta Aggiornamento delle configurazioni dei componenti.

Console

```
{
  "mqttTopicMapping": {
    "HelloWorldIotCore": {
      "topic": "clients/+/hello/world",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

AWS CLI

```
{
  "components": {
    "aws.greengrass.clientdevices.mqtt.Bridge": {
      "version": "2.0.0",
      "configurationUpdate": {
        "merge": "{\"mqttTopicMapping\":{\"HelloWorldIotCore\":{\"topic\":\"clients/
+/hello/world\",\"source\":\"LocalMqtt\",\"target\":\"IotCore\"}}}"
      }
    }
    ...
  }
}
```

Inoltra messaggi MQTT

Per inoltrare messaggi MQTT tra dispositivi client eAWS IoT Core, configura e distribuisci il componente MQTT Bridge e specifica gli argomenti da inoltrare.

Example Esempio: inoltrare messaggi su un argomento dai dispositivi client a AWS IoT Core

La seguente configurazione del componente bridge MQTT specifica l'inoltro dei messaggi su argomenti che corrispondono al filtro degli argomenti dai `clients/+/hello/world/event` dispositivi client a. AWS IoT Core

```
{
  "mqttTopicMapping": {
    "HelloWorldEvent": {
      "topic": "clients/+/hello/world/event",
      "source": "LocalMqtt",
      "target": "IotCore"
    }
  }
}
```

Example Esempio: inoltrare messaggi su un argomento dai dispositivi client AWS IoT Core

La seguente configurazione del componente bridge MQTT specifica l'inoltro dei messaggi su argomenti che corrispondono al filtro degli `clients/+/hello/world/event/response` argomenti ai dispositivi client. AWS IoT Core

```
{
  "mqttTopicMapping": {
    "HelloWorldEventConfirmation": {
      "topic": "clients/+/hello/world/event/response",
      "source": "IotCore",
      "target": "LocalMqtt"
    }
  }
}
```

Interagisci con i dispositivi client nei componenti

Puoi sviluppare componenti Greengrass personalizzati che interagiscono con i dispositivi client collegati a un dispositivo principale. Ad esempio, puoi sviluppare componenti che eseguono le seguenti operazioni:

- Agisci sui messaggi MQTT dai dispositivi client e invia dati alle Cloud AWS destinazioni.
- Invia messaggi MQTT ai dispositivi client per avviare azioni.

I dispositivi client si connettono e comunicano con un dispositivo principale tramite il componente broker MQTT che viene eseguito sul dispositivo principale. Per impostazione predefinita, i dispositivi client possono comunicare tra loro solo tramite MQTT e i componenti Greengrass non possono ricevere questi messaggi MQTT o inviare messaggi ai dispositivi client.

I componenti Greengrass utilizzano l'interfaccia di pubblicazione/sottoscrizione locale per comunicare su un dispositivo principale. Per comunicare con i dispositivi client nei componenti Greengrass, configurate il componente bridge MQTT per eseguire le seguenti operazioni:

- Inoltra i messaggi MQTT dai dispositivi client alla pubblicazione/sottoscrizione locale.
- Inoltra i messaggi MQTT dalla pubblicazione/sottoscrizione locali ai dispositivi client.

Puoi anche interagire con le ombre dei dispositivi client nei componenti Greengrass. Per ulteriori informazioni, consulta Interazione e sincronizzazione delle ombre dei dispositivi client.

Argomenti

Configura e distribuisci il componente bridge MQTT

Il componente bridge MQTT utilizza un elenco di mappature di argomenti, ciascuna delle quali specifica un'origine e una destinazione del messaggio. Per comunicare con i dispositivi client, implementate il componente bridge MQTT e specificate ogni argomento di origine e destinazione nella configurazione del componente.

Per distribuire il componente bridge MQTT su un dispositivo principale o su un gruppo di dispositivi principali, create una distribuzione che includa il componente. `aws.greengrass.clientdevices.mqtt.Bridge` Specificate le mappature degli argomenti nella configurazione `mqttTopicMapping` del componente bridge MQTT nella distribuzione.

L'esempio seguente definisce una distribuzione che configura il componente bridge MQTT per inoltrare l'`clients/MyClientDevice1/hello/world`argomento dai dispositivi client al broker di pubblicazione/sottoscrizione locale. L'aggiornamento della `merge` configurazione richiede un oggetto JSON serializzato. Per ulteriori informazioni, consulta Aggiornamento delle configurazioni dei componenti.

Console

```
{
  "mqttTopicMapping": {
    "HelloWorldPubsub": {
      "topic": "clients/MyClientDevice1/hello/world",
      "source": "LocalMqtt",
      "target": "Pubsub"
    }
  }
}
```

AWS CLI

```
{
  "components": {
    "aws.greengrass.clientdevices.mqtt.Bridge": {
      "version": "2.0.0",
      "configurationUpdate": {
        "merge": "\"mqttTopicMapping\":{\"HelloWorldPubsub\":{\"topic\":\"clients/
MyClientDevice1/hello/world\",\"source\":\"LocalMqtt\",\"target\":\"Pubsub\"}}}"
      }
    }
    ...
  }
}
```

È possibile utilizzare i caratteri jolly degli argomenti MQTT per inoltrare messaggi su argomenti che corrispondono a un filtro di argomento. Se si utilizza MQTT bridge v2.2.0 o versione successiva, è possibile utilizzare i caratteri jolly degli argomenti MQTT nei filtri degli argomenti quando il broker di origine è locale/publish/subscribe. Per ulteriori informazioni, vedete Configurazione dei componenti del bridge MQTT.

Ricevere messaggi MQTT dai dispositivi client

È possibile abbonarsi agli argomenti di pubblicazione/sottoscrizione locali configurati per il componente bridge MQTT per ricevere messaggi dai dispositivi client.

Per ricevere messaggi MQTT dai dispositivi client in componenti personalizzati

1. Configurate e implementate il componente bridge MQTT per inoltrare i messaggi da un argomento MQTT in cui i dispositivi client vengono pubblicati su un argomento di pubblicazione/sottoscrizione locale.
2. Utilizzate l'interfaccia IPC di pubblicazione/sottoscrizione locale per sottoscrivere l'argomento in cui il bridge MQTT inoltra i messaggi. Per ulteriori informazioni, consultare Pubblicare/sottoscrivere messaggi locali e SubscribeToTopic.

Il tutorial Connect and test client devices include una sezione in cui si sviluppa un componente che sottoscrive i messaggi da un dispositivo client. Per ulteriori informazioni, consulta Fase 4: Sviluppare un componente che comunichi con i dispositivi client.

Inviare messaggi MQTT ai dispositivi client

È possibile pubblicare negli argomenti di pubblicazione/sottoscrizione locali configurati per il componente bridge MQTT per l'invio di messaggi ai dispositivi client.

Per pubblicare messaggi MQTT su dispositivi client in componenti personalizzati

1. Configurate e implementate il componente bridge MQTT per inoltrare i messaggi da un argomento di pubblicazione/sottoscrizione locale a un argomento MQTT a cui i dispositivi client effettuano la sottoscrizione.
2. Utilizzate l'interfaccia IPC di pubblicazione/sottoscrizione locale per la pubblicazione sull'argomento in cui il bridge MQTT inoltra i messaggi. Per ulteriori informazioni, consultare Pubblicare/sottoscrivere messaggi locali e PublishToTopic.

Interazione e sincronizzazione delle ombre dei dispositivi client

È possibile utilizzare il componente shadow manager per gestire le ombre locali, incluse le ombre dei dispositivi client. È possibile utilizzare shadow manager per effettuare le seguenti operazioni:

- Interagisci con le ombre dei dispositivi client nei componenti Greengrass.
- Sincronizza le ombre dei dispositivi client con. AWS IoT Core

> **Note**
>
> Per impostazione predefinita, il componente Shadow Manager non sincronizza le ombre conAWS IoT Core. È necessario configurare il componente shadow manager per specificare con quali ombre del dispositivo client sincronizzare.

Argomenti

Prerequisiti

Per interagire con le ombre dei dispositivi client e sincronizzare le ombre dei dispositivi clientAWS IoT Core, un dispositivo principale deve soddisfare i seguenti requisiti:

- Il dispositivo principale deve eseguire i seguenti componenti, oltre ai componenti Greengrass per il supporto dei dispositivi client:
  - Greengrass nucleus v2.6.0 o successivo
  - Shadow manager v2.2.0 o successivo
  - MQTT bridge v2.2.0 o successivo
- Il componente di autenticazione del dispositivo client deve essere configurato per consentire ai dispositivi client di comunicare su argomenti relativi ai dispositivi shadow.

Abilita Shadow Manager per comunicare con i dispositivi client

Per impostazione predefinita, il componente shadow manager non gestisce le ombre dei dispositivi client. Per abilitare questa funzionalità, è necessario inoltrare i messaggi MQTT tra i dispositivi client e il componente shadow manager. I dispositivi client utilizzano i messaggi MQTT per ricevere e inviare aggiornamenti shadow dei dispositivi. Il componente shadow manager sottoscrive l'interfaccia locale di pubblicazione/sottoscrizione di Greengrass, quindi è possibile configurare il componente bridge MQTT per inoltrare messaggi MQTT su argomenti shadow del dispositivo.

Il componente bridge MQTT utilizza un elenco di mappature di argomenti, ciascuna delle quali specifica un'origine e una destinazione del messaggio. Per abilitare il componente shadow manager alla gestione delle ombre dei dispositivi client, implementate il componente bridge MQTT e specificate gli argomenti shadow per le ombre dei dispositivi client. È necessario configurare il bridge per inoltrare i messaggi in entrambe le direzioni tra MQTT locale e pubblicazione/sottoscrizione locale.

Per distribuire il componente bridge MQTT su un dispositivo principale o su un gruppo di dispositivi principali, create una distribuzione che includa il componente. `aws.greengrass.clientdevices.mqtt.Bridge` Specificate le mappature degli argomenti nella configurazione `mqttTopicMapping` del componente bridge MQTT nella distribuzione.

Utilizzate i seguenti esempi per configurare il componente bridge MQTT per abilitare la comunicazione tra i dispositivi client e il componente shadow manager.

> **Note**
>
> È possibile utilizzare questi esempi di configurazione nella AWS IoT Greengrass console. Se utilizzi l'AWS IoT GreengrassAPI, l'aggiornamento della `merge` configurazione richiede un oggetto JSON serializzato, quindi devi serializzare i seguenti oggetti JSON in stringhe. Per ulteriori informazioni, consulta Aggiornamento delle configurazioni dei componenti.

Example Esempio: gestione di tutte le ombre dei dispositivi client

Il seguente esempio di configurazione del bridge MQTT consente a shadow manager di gestire tutte le ombre per tutti i dispositivi client.

```
{
  "mqttTopicMapping": {
    "ShadowsLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/#",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
```

```
}
```

Example Esempio: gestione delle ombre per un dispositivo client

Il seguente esempio di configurazione del bridge MQTT consente a shadow manager di gestire tutte le ombre per un dispositivo client denominato. `MyClientDevice`

```
{
  "mqttTopicMapping": {
    "ShadowsLocalMqttToPubsub": {
      "topic": "$aws/things/MyClientDevice/shadow/#",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsPubsubToLocalMqtt": {
      "topic": "$aws/things/MyClientDevice/shadow/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
}
```

Example Esempio: gestire un'ombra denominata per tutti i dispositivi client

Il seguente esempio di configurazione di bridge MQTT consente a shadow manager di gestire uno shadow denominato `DeviceConfiguration` per tutti i dispositivi client.

```
{
  "mqttTopicMapping": {
    "ShadowsLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/name/DeviceConfiguration/#",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "ShadowsPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/name/DeviceConfiguration/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
}
```

Example Esempio: gestione delle ombre senza nome di tutti i dispositivi client

Il seguente esempio di configurazione del bridge MQTT consente a shadow manager di gestire ombre senza nome, ma non ombre con nome, per tutti i dispositivi client.

```
{
  "mqttTopicMapping": {
    "DeleteShadowLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/delete",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "DeleteShadowPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/delete/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    },
    "GetShadowLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/get",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "GetShadowPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/get/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    },
    "UpdateShadowLocalMqttToPubsub": {
      "topic": "$aws/things/+/shadow/update",
      "source": "LocalMqtt",
      "target": "Pubsub"
    },
    "UpdateShadowPubsubToLocalMqtt": {
      "topic": "$aws/things/+/shadow/update/#",
      "source": "Pubsub",
      "target": "LocalMqtt"
    }
  }
}
```

Interagisci con le ombre dei dispositivi client nei componenti

È possibile sviluppare componenti personalizzati che utilizzano il servizio shadow locale per leggere e modificare i documenti shadow locali dei dispositivi client. Per ulteriori informazioni, consulta Interagisci con le ombre nei componenti.

Sincronizza le ombre dei dispositivi client con AWS IoT Core

È possibile configurare il componente shadow manager per sincronizzare gli stati shadow dei dispositivi client locali con. AWS IoT Core Per ulteriori informazioni, consulta Sincronizza le ombre del dispositivo locale con AWS IoT Core.

Risoluzione dei problemi relativi ai dispositivi client

Utilizza le informazioni e le soluzioni per la risoluzione dei problemi in questa sezione per risolvere i problemi con i dispositivi client Greengrass e i componenti dei dispositivi client.

Argomenti

Problemi relativi alla scoperta di Greengrass

Utilizza le seguenti informazioni per risolvere i problemi relativi a Greengrass discovery. Questi problemi possono verificarsi quando i dispositivi client utilizzano l'API Greengrass discovery per identificare un dispositivo principale Greengrass a cui connettersi.

Argomenti

Problemi relativi alla scoperta di Greengrass (API HTTP)

Utilizza le seguenti informazioni per risolvere i problemi relativi a Greengrass discovery. Potresti riscontrare questi errori se provi l'API di rilevamento con cURL.

Argomenti

curl: (52) Empty reply from server

Potresti visualizzare questo errore se specifichi un AWS IoT certificato inattivo nella richiesta.

Verifica che il dispositivo client abbia un certificato allegato e che il certificato sia attivo. Per ulteriori informazioni, consulta Allegare un elemento o una politica a un certificato client e Attivare o disattivare un certificato client nella Guida per gli AWS IoT Core sviluppatori.

HTTP 403: {"message":null,"traceId":"a1b2c3d4-5678-90ab-cdef-11111EXAMPLE"}

Potresti visualizzare questo errore se il dispositivo client non dispone dell'autorizzazione `greengrass:Discover` per effettuare chiamate autonomamente.

Verifica che il certificato del dispositivo client disponga di una politica che lo consenta`greengrass:Discover`. Non puoi usare thing policy variables (`iot:Connection.Thing.*`) nella `Resource` sezione relativa a questa autorizzazione. Per ulteriori informazioni, consulta Autenticazione e autorizzazione Discovery.

HTTP 404: {"errorMessage":"The thing provided for discovery was not found"}

Potresti visualizzare questo errore nei seguenti casi:

- Il dispositivo client non è associato a nessun dispositivo o AWS IoT Greengrass V1 gruppo Greengrass core.
- Nessuno dei dispositivi o AWS IoT Greengrass V1 gruppi principali Greengrass associati al dispositivo client dispone di un endpoint broker MQTT.
- Nessuno dei dispositivi core Greengrass associati al dispositivo client esegue il componente di autenticazione del dispositivo client.

Verifica che il dispositivo client sia associato al dispositivo principale a cui desideri collegarlo. Quindi, verifica che il dispositivo principale esegua il componente di autenticazione del dispositivo client e disponga di almeno un endpoint del broker MQTT. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Associa i dispositivi client
- Gestisci gli endpoint principali dei dispositivi
- Configura cloud discovery (console)

Problemi di scoperta di Greengrass (SDK per dispositivi AWS IoTv2 per Python)

Usa le seguenti informazioni per risolvere i problemi relativi alla scoperta di Greengrass nella versione 2 per Python. SDK per dispositivi AWS IoT

Argomenti

- awscrt.exceptions.AwsCrtError: AWS_ERROR_HTTP_CONNECTION_CLOSED: The connection has closed or is closing.
- awsiot.greengrass_discovery.DiscoveryException: ('Error during discover call: response_code=403', 403)
- awsiot.greengrass_discovery.DiscoveryException: ('Error during discover call: response_code=404', 404)

awscrt.exceptions.AwsCrtError: AWS_ERROR_HTTP_CONNECTION_CLOSED: The connection has closed or is closing.

Potresti visualizzare questo errore se specifichi un certificato inattivo nella AWS IoT richiesta.

Verifica che il dispositivo client abbia un certificato allegato e che il certificato sia attivo. Per ulteriori informazioni, consulta Allegare un elemento o una politica a un certificato client e Attivare o disattivare un certificato client nella Guida per gli AWS IoT Core sviluppatori.

awsiot.greengrass_discovery.DiscoveryException: ('Error during discover call: response_code=403', 403)

Potresti visualizzare questo errore se il dispositivo client non dispone dell'autorizzazione `greengrass:Discover` per effettuare chiamate autonomamente.

Verifica che il certificato del dispositivo client disponga di una politica che lo consentagreengrass:Discover. Non puoi usare thing policy variables (iot:Connection.Thing.*) nella Resource sezione relativa a questa autorizzazione. Per ulteriori informazioni, consulta Autenticazione e autorizzazione Discovery.

awsiot.greengrass_discovery.DiscoveryException: ('Error during discover call: response_code=404', 404)

Potresti visualizzare questo errore nei seguenti casi:

- Il dispositivo client non è associato a nessun dispositivo o AWS IoT Greengrass V1 gruppo Greengrass core.
- Nessuno dei dispositivi o AWS IoT Greengrass V1 gruppi principali Greengrass associati al dispositivo client dispone di un endpoint broker MQTT.
- Nessuno dei dispositivi core Greengrass associati al dispositivo client esegue il componente di autenticazione del dispositivo client.

Verifica che il dispositivo client sia associato al dispositivo principale a cui desideri collegarlo. Quindi, verifica che il dispositivo principale esegua il componente di autenticazione del dispositivo client e disponga di almeno un endpoint del broker MQTT. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Associa i dispositivi client
- Gestisci gli endpoint principali dei dispositivi
- Configura cloud discovery (console)

Problemi relativi alla scoperta di Greengrass (SDK per dispositivi AWS IoTv2 per C++)

Utilizza le seguenti informazioni per risolvere i problemi relativi a Greengrass discovery nella versione 2 per C++. SDK per dispositivi AWS IoT

Argomenti

- aws-c-http: AWS_ERROR_HTTP_CONNECTION_CLOSED, The connection has closed or is closing.
- aws-c-common: AWS_ERROR_UNKNOWN, Unknown error. (HTTP 403)
- aws-c-common: AWS_ERROR_UNKNOWN, Unknown error. (HTTP 404)

aws-c-http: AWS_ERROR_HTTP_CONNECTION_CLOSED, The connection has closed or is closing.

Potresti visualizzare questo errore se specifichi un certificato inattivo nella AWS IoT richiesta.

Verifica che il dispositivo client abbia un certificato allegato e che il certificato sia attivo. Per ulteriori informazioni, consulta Allegare un elemento o una politica a un certificato client e Attivare o disattivare un certificato client nella Guida per gli AWS IoT Core sviluppatori.

aws-c-common: AWS_ERROR_UNKNOWN, Unknown error. (HTTP 403)

Potresti visualizzare questo errore se il dispositivo client non dispone dell'autorizzazione `greengrass:Discover` per effettuare chiamate autonomamente.

Verifica che il certificato del dispositivo client disponga di una politica che lo consentagreengrass:Discover. Non puoi usare thing policy variables (`iot:Connection.Thing.*`) nella `Resource` sezione relativa a questa autorizzazione. Per ulteriori informazioni, consulta Autenticazione e autorizzazione Discovery.

aws-c-common: AWS_ERROR_UNKNOWN, Unknown error. (HTTP 404)

Potresti visualizzare questo errore nei seguenti casi:

- Il dispositivo client non è associato a nessun dispositivo o AWS IoT Greengrass V1 gruppo Greengrass core.
- Nessuno dei dispositivi o AWS IoT Greengrass V1 gruppi principali Greengrass associati al dispositivo client dispone di un endpoint broker MQTT.
- Nessuno dei dispositivi core Greengrass associati al dispositivo client esegue il componente di autenticazione del dispositivo client.

Verifica che il dispositivo client sia associato al dispositivo principale a cui desideri collegarlo. Quindi, verifica che il dispositivo principale esegua il componente di autenticazione del dispositivo client e disponga di almeno un endpoint del broker MQTT. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Associa i dispositivi client
- Gestisci gli endpoint principali dei dispositivi
- Configura cloud discovery (console)

Problemi relativi alla scoperta di Greengrass (SDK per dispositivi AWS IoTv2 per) JavaScript

Utilizza le seguenti informazioni per risolvere i problemi relativi a Greengrass discovery nella versione 2 per. SDK per dispositivi AWS IoT JavaScript

Argomenti

Error: aws-c-http: AWS_ERROR_HTTP_CONNECTION_CLOSED, The connection has closed or is closing.

Potresti visualizzare questo errore se specifichi un certificato inattivo nella richiestaAWS IoT.

Verifica che il dispositivo client abbia un certificato allegato e che il certificato sia attivo. Per ulteriori informazioni, consulta Allegare un elemento o una politica a un certificato client e Attivare o disattivare un certificato client nella Guida per gli AWS IoT Core sviluppatori.

Error: Discovery failed (headers: [object Object]) { response_code: 403 }

Potresti visualizzare questo errore se il dispositivo client non dispone dell'autorizzazione `greengrass:Discover` per effettuare chiamate autonomamente.

Verifica che il certificato del dispositivo client disponga di una politica che lo consenta`greengrass:Discover`. Non puoi usare thing policy variables (`iot:Connection.Thing.*`) nella `Resource` sezione relativa a questa autorizzazione. Per ulteriori informazioni, consulta Autenticazione e autorizzazione Discovery.

Error: Discovery failed (headers: [object Object]) { response_code: 404 }

Potresti visualizzare questo errore nei seguenti casi:

- Il dispositivo client non è associato a nessun dispositivo o AWS IoT Greengrass V1 gruppo Greengrass core.

- Nessuno dei dispositivi o AWS IoT Greengrass V1 gruppi principali Greengrass associati al dispositivo client dispone di un endpoint broker MQTT.
- Nessuno dei dispositivi core Greengrass associati al dispositivo client esegue il componente di autenticazione del dispositivo client.

Verifica che il dispositivo client sia associato al dispositivo principale a cui desideri collegarlo. Quindi, verifica che il dispositivo principale esegua il componente di autenticazione del dispositivo client e disponga di almeno un endpoint del broker MQTT. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Associa i dispositivi client
- Gestisci gli endpoint principali dei dispositivi
- Configura cloud discovery (console)

Error: Discovery failed (headers: [object Object])

Potresti visualizzare questo errore (senza un codice di risposta HTTP) quando esegui l'esempio Greengrass discovery. Questo errore può verificarsi per diversi motivi.

- Potresti visualizzare questo errore se il dispositivo client non dispone dell'autorizzazione `greengrass:Discover` per effettuare chiamate autonomamente.

  Verifica che il certificato del dispositivo client disponga di una politica che lo consenta`greengrass:Discover`. Non puoi usare thing policy variables (`iot:Connection.Thing.*`) nella `Resource` sezione relativa a questa autorizzazione. Per ulteriori informazioni, consulta Autenticazione e autorizzazione Discovery.
- Potresti visualizzare questo errore nei seguenti casi:
  - Il dispositivo client non è associato a nessun dispositivo o AWS IoT Greengrass V1 gruppo Greengrass core.
  - Nessuno dei dispositivi o AWS IoT Greengrass V1 gruppi principali Greengrass associati al dispositivo client dispone di un endpoint broker MQTT.
  - Nessuno dei dispositivi core Greengrass associati al dispositivo client esegue il componente di autenticazione del dispositivo client.

  Verifica che il dispositivo client sia associato al dispositivo principale a cui desideri collegarlo. Quindi, verifica che il dispositivo principale esegua il componente di autenticazione del dispositivo

client e disponga di almeno un endpoint del broker MQTT. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Associa i dispositivi client
- Gestisci gli endpoint principali dei dispositivi
- Configura cloud discovery (console)

Problemi di scoperta di Greengrass (SDK per dispositivi AWS IoTv2 per Java)

Utilizza le seguenti informazioni per risolvere i problemi relativi a Greengrass discovery nella versione 2 per Java. SDK per dispositivi AWS IoT

Argomenti

- software.amazon.awssdk.crt.CrtRuntimeException: Error Getting Response Status Code from HttpStream. (aws_last_error: AWS_ERROR_HTTP_DATA_NOT_AVAILABLE(2062), This data is not yet available.)
- java.lang.RuntimeException: Error x-amzn-ErrorType(403)
- java.lang.RuntimeException: Error x-amzn-ErrorType(404)

software.amazon.awssdk.crt.CrtRuntimeException: Error Getting Response Status Code from HttpStream. (aws_last_error: AWS_ERROR_HTTP_DATA_NOT_AVAILABLE(2062), This data is not yet available.)

Potresti visualizzare questo errore se specifichi un certificato inattivo nella richiestaAWS IoT.

Verifica che il dispositivo client abbia un certificato allegato e che il certificato sia attivo. Per ulteriori informazioni, consulta Allegare un elemento o una politica a un certificato client e Attivare o disattivare un certificato client nella Guida per gli AWS IoT Core sviluppatori.

java.lang.RuntimeException: Error x-amzn-ErrorType(403)

Potresti visualizzare questo errore se il dispositivo client non dispone dell'autorizzazione `greengrass:Discover` per effettuare chiamate autonomamente.

Verifica che il certificato del dispositivo client disponga di una politica che lo consentagreengrass:Discover. Non puoi usare thing policy variables (`iot:Connection.Thing.*`) nella `Resource` sezione relativa a questa autorizzazione. Per ulteriori informazioni, consulta Autenticazione e autorizzazione Discovery.

java.lang.RuntimeException: Error x-amzn-ErrorType(404)

Potresti visualizzare questo errore nei seguenti casi:

- Il dispositivo client non è associato a nessun dispositivo o AWS IoT Greengrass V1 gruppo Greengrass core.
- Nessuno dei dispositivi o AWS IoT Greengrass V1 gruppi principali Greengrass associati al dispositivo client dispone di un endpoint broker MQTT.
- Nessuno dei dispositivi core Greengrass associati al dispositivo client esegue il componente di autenticazione del dispositivo client.

Verifica che il dispositivo client sia associato al dispositivo principale a cui desideri collegarlo. Quindi, verifica che il dispositivo principale esegua il componente di autenticazione del dispositivo client e disponga di almeno un endpoint del broker MQTT. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Associa i dispositivi client
- Gestisci gli endpoint principali dei dispositivi
- Configura cloud discovery (console)

Problemi di connessione MQTT

Utilizza le seguenti informazioni per risolvere i problemi relativi alle connessioni MQTT dei dispositivi client. Questi problemi possono verificarsi quando i dispositivi client tentano di connettersi a un dispositivo principale tramite MQTT.

Argomenti

- io.moquette.broker.Authorizator: Client does not have read permissions on the topic
- Problemi di connessione MQTT (Python)
- Problemi di connessione MQTT (C++)
- Problemi di connessione MQTT (Java)
- Problemi di connessione MQTT () JavaScript

io.moquette.broker.Authorizator: Client does not have read permissions on the topic

Potresti visualizzare questo errore nei log di Greengrass quando un dispositivo client tenta di iscriversi a un argomento MQTT per il quale non dispone dell'autorizzazione. Il messaggio di errore include l'argomento.

Verificate che la configurazione del componente di autenticazione del dispositivo client includa quanto segue:

- Un gruppo di dispositivi che corrisponde al dispositivo client.
- Una politica di autorizzazione dei dispositivi client per quel gruppo di dispositivi che concede l'`mqtt:subscribe`autorizzazione per l'argomento.

Per ulteriori informazioni su come distribuire e configurare il componente di autenticazione del dispositivo client, consulta quanto segue:

- Configura cloud discovery (console)
- Autenticazione del dispositivo client
- Creare distribuzione

Problemi di connessione MQTT (Python)

Usa le seguenti informazioni per risolvere i problemi relativi alle connessioni MQTT dei dispositivi client quando usi la versione SDK per dispositivi AWS IoT2 per Python.

Argomenti

- AWS_ERROR_MQTT_PROTOCOL_ERROR: Protocol error occurred
- AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

AWS_ERROR_MQTT_PROTOCOL_ERROR: Protocol error occurred

Potresti visualizzare questo errore se il componente di autenticazione del dispositivo client non definisce una politica di autorizzazione del dispositivo client che concede al dispositivo client l'autorizzazione alla connessione.

Verifica che la configurazione del componente di autenticazione del dispositivo client includa quanto segue:

- Un gruppo di dispositivi che corrisponde al dispositivo client.
- Una politica di autorizzazione dei dispositivi client per quel gruppo di dispositivi che concede l'`mqtt:connect`autorizzazione per il dispositivo client.

Per ulteriori informazioni su come distribuire e configurare il componente di autenticazione del dispositivo client, consulta quanto segue:

- Configura cloud discovery (console)
- Autenticazione del dispositivo client
- Creare distribuzione

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Potresti visualizzare questo errore se il componente di autenticazione del dispositivo client non definisce una politica di autorizzazione del dispositivo client che conceda al dispositivo client l'autorizzazione alla connessione.

Verifica che la configurazione del componente di autenticazione del dispositivo client includa quanto segue:

- Un gruppo di dispositivi che corrisponde al dispositivo client.
- Una politica di autorizzazione dei dispositivi client per quel gruppo di dispositivi che concede l'`mqtt:connect`autorizzazione per il dispositivo client.

Per ulteriori informazioni su come distribuire e configurare il componente di autenticazione del dispositivo client, consulta quanto segue:

- Configura cloud discovery (console)
- Autenticazione del dispositivo client
- Creare distribuzione

Problemi di connessione MQTT (C++)

Utilizzate le seguenti informazioni per risolvere i problemi relativi alle connessioni MQTT dei dispositivi client quando utilizzate la versione v2 per C++. SDK per dispositivi AWS IoT

Argomenti

AWS_ERROR_MQTT_PROTOCOL_ERROR: Protocol error occurred

Potresti visualizzare questo errore se il componente di autenticazione del dispositivo client non definisce una politica di autorizzazione del dispositivo client che concede al dispositivo client l'autorizzazione alla connessione.

Verifica che la configurazione del componente di autenticazione del dispositivo client includa quanto segue:

- Un gruppo di dispositivi che corrisponde al dispositivo client.
- Una politica di autorizzazione dei dispositivi client per quel gruppo di dispositivi che concede l'`mqtt:connect`autorizzazione per il dispositivo client.

Per ulteriori informazioni su come distribuire e configurare il componente di autenticazione del dispositivo client, consulta quanto segue:

- Configura cloud discovery (console)
- Autenticazione del dispositivo client
- Creare distribuzione

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Potresti visualizzare questo errore se il componente di autenticazione del dispositivo client non definisce una politica di autorizzazione del dispositivo client che conceda al dispositivo client l'autorizzazione alla connessione.

Verifica che la configurazione del componente di autenticazione del dispositivo client includa quanto segue:

- Un gruppo di dispositivi che corrisponde al dispositivo client.
- Una politica di autorizzazione dei dispositivi client per quel gruppo di dispositivi che concede l'`mqtt:connect`autorizzazione per il dispositivo client.

Per ulteriori informazioni su come distribuire e configurare il componente di autenticazione del dispositivo client, consulta quanto segue:

- Configura cloud discovery (console)
- Autenticazione del dispositivo client
- Creare distribuzione

Problemi di connessione MQTT (Java)

Utilizzate le seguenti informazioni per risolvere i problemi relativi alle connessioni MQTT dei dispositivi client quando utilizzate la SDK per dispositivi AWS IoT versione v2 per Java.

Argomenti

- software.amazon.awssdk.crt.mqtt.MqttException: Protocol error occurred
- AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

software.amazon.awssdk.crt.mqtt.MqttException: Protocol error occurred

Potresti visualizzare questo errore se il componente di autenticazione del dispositivo client non definisce una politica di autorizzazione del dispositivo client che concede al dispositivo client l'autorizzazione alla connessione.

Verifica che la configurazione del componente di autenticazione del dispositivo client includa quanto segue:

- Un gruppo di dispositivi che corrisponde al dispositivo client.
- Una politica di autorizzazione dei dispositivi client per quel gruppo di dispositivi che concede l'`mqtt:connect`autorizzazione per il dispositivo client.

Per ulteriori informazioni su come distribuire e configurare il componente di autenticazione del dispositivo client, consulta quanto segue:

- Configura cloud discovery (console)
- Autenticazione del dispositivo client
- Creare distribuzione

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Potresti visualizzare questo errore se il componente di autenticazione del dispositivo client non definisce una politica di autorizzazione del dispositivo client che conceda al dispositivo client l'autorizzazione alla connessione.

Verifica che la configurazione del componente di autenticazione del dispositivo client includa quanto segue:

- Un gruppo di dispositivi che corrisponde al dispositivo client.
- Una politica di autorizzazione dei dispositivi client per quel gruppo di dispositivi che concede l'`mqtt:connect`autorizzazione per il dispositivo client.

Per ulteriori informazioni su come distribuire e configurare il componente di autenticazione del dispositivo client, consulta quanto segue:

- Configura cloud discovery (console)
- Autenticazione del dispositivo client
- Creare distribuzione

Problemi di connessione MQTT () JavaScript

Utilizzate le seguenti informazioni per risolvere i problemi relativi alle connessioni MQTT dei dispositivi client quando utilizzate la versione v2 per. SDK per dispositivi AWS IoT JavaScript

Argomenti

AWS_ERROR_MQTT_PROTOCOL_ERROR: Protocol error occurred

Potresti visualizzare questo errore se il componente di autenticazione del dispositivo client non definisce una politica di autorizzazione del dispositivo client che conceda al dispositivo client l'autorizzazione alla connessione.

Verifica che la configurazione del componente di autenticazione del dispositivo client includa quanto segue:

- Un gruppo di dispositivi che corrisponde al dispositivo client.
- Una politica di autorizzazione dei dispositivi client per quel gruppo di dispositivi che concede l'`mqtt:connect`autorizzazione per il dispositivo client.

Per ulteriori informazioni su come distribuire e configurare il componente di autenticazione del dispositivo client, consulta quanto segue:

- Configura cloud discovery (console)
- Autenticazione del dispositivo client
- Creare distribuzione

AWS_ERROR_MQTT_UNEXPECTED_HANGUP: Unexpected hangup occurred

Potresti visualizzare questo errore se il componente di autenticazione del dispositivo client non definisce una politica di autorizzazione del dispositivo client che conceda al dispositivo client l'autorizzazione alla connessione.

Verifica che la configurazione del componente di autenticazione del dispositivo client includa quanto segue:

- Un gruppo di dispositivi che corrisponde al dispositivo client.
- Una politica di autorizzazione dei dispositivi client per quel gruppo di dispositivi che concede l'`mqtt:connect`autorizzazione per il dispositivo client.

Per ulteriori informazioni su come distribuire e configurare il componente di autenticazione del dispositivo client, consulta quanto segue:

- Configura cloud discovery (console)
- Autenticazione del dispositivo client
- Creare distribuzione

Interagisci con le ombre dei dispositivi

I dispositivi core Greengrass possono interagire con le ombre dei dispositivi AWS IoT utilizzando i componenti. Un'ombra è un documento JSON che memorizza le informazioni sullo stato corrente o desiderato per qualsiasi cosa. AWS IoT Le ombre possono rendere disponibile lo stato di un dispositivo ad altri AWS IoT Greengrass componenti, a prescindere dal fatto che il dispositivo sia connesso AWS IoT o meno. Ogni AWS IoT dispositivo ha la propria ombra classica senza nome. Puoi anche creare più ombre con nome per ogni dispositivo.

I dispositivi e i servizi possono creare, aggiornare ed eliminare cloud shadow utilizzando MQTT e gli argomenti shadow MQTT riservati, HTTP utilizzando l'API REST Device Shadow e for. AWS CLIAWS IoT

Il componente shadow manager consente ai componenti Greengrass di creare, aggiornare ed eliminare le ombre locali utilizzando il servizio shadow locale e gli argomenti shadow locali di pubblicazione/sottoscrizione. Lo shadow manager gestisce anche l'archiviazione di questi documenti shadow locali sul dispositivo principale e gestisce la sincronizzazione delle informazioni sullo stato dell'ombra con le ombre del cloud.

È inoltre possibile utilizzare il componente shadow manager per gestire le shadow locali per i dispositivi client che si connettono al dispositivo principale. Per consentire allo shadow manager di gestire le ombre dei dispositivi client, è necessario configurare il componente bridge MQTT per inoltrare i messaggi tra il broker MQTT locale e il servizio di pubblicazione/sottoscrizione locale. Per ulteriori informazioni, consulta Interazione e sincronizzazione delle ombre dei dispositivi client.

Per ulteriori informazioni sui concetti relativi al AWS IoT device shadow, consulta AWS IoTDevice Shadow service nella AWS IoTDeveloper Guide.

Argomenti

- Interagisci con le ombre nei componenti
- Sincronizza le ombre del dispositivo locale con AWS IoT Core

Interagisci con le ombre nei componenti

È possibile sviluppare componenti personalizzati, inclusi i componenti della funzione Lambda, che utilizzano il servizio shadow locale per leggere e modificare documenti shadow locali e documenti shadow dei dispositivi client.

I componenti personalizzati interagiscono con il servizio shadow locale utilizzando le librerie IPC AWS IoT Greengrass Core di. SDK per dispositivi AWS IoT Il componente shadow manager abilita il servizio shadow locale sul dispositivo principale.

Per distribuire il componente shadow manager su un dispositivo principale Greengrass, crea una distribuzione che `aws.greengrass.ShadowManager` includa il componente.

> **Note**
>
> Per impostazione predefinita, la distribuzione del componente shadow manager abilita solo le operazioni shadow locali. AWS IoT GreengrassPer abilitare la sincronizzazione delle informazioni sullo stato dello shadow per le ombre dei dispositivi principali o qualsiasi ombra per i dispositivi client con i corrispondenti documenti cloud shadow inAWS IoT Core, è necessario creare un aggiornamento della configurazione per il componente shadow manager che include il parametro. `synchronize` Per ulteriori informazioni, consulta Sincronizza le ombre del dispositivo locale con AWS IoT Core.

Argomenti

- Recuperate e modificate gli stati delle ombre
- Reagisci ai cambiamenti dello stato ombra

Recuperate e modificate gli stati delle ombre

Le operazioni IPC shadow recuperano e aggiornano le informazioni sullo stato nei documenti shadow locali. Il componente shadow manager gestisce l'archiviazione di questi documenti shadow sul dispositivo principale.

Per modificare gli stati shadow locali

1. Aggiungi politiche di autorizzazione alla ricetta del tuo componente personalizzato per consentire al componente di ricevere messaggi su argomenti shadow locali.

   Ad esempio, le politiche di autorizzazione, vedi Esempi di politiche di autorizzazione IPC Shadow Local.
2. Utilizzate le operazioni shadow IPC per recuperare e modificare le informazioni sullo stato dello shadow. Per ulteriori informazioni sull'utilizzo delle operazioni IPC shadow nel codice dei componenti, vedere. Interagisci con le ombre locali

> **Note**
>
> Per consentire a un dispositivo principale di interagire con le ombre dei dispositivi client, è inoltre necessario configurare e distribuire il componente bridge MQTT. Per ulteriori informazioni, consultate Attivare lo shadow manager per comunicare con i dispositivi client.

Reagisci ai cambiamenti dello stato ombra

I componenti Greengrass utilizzano l'interfaccia di pubblicazione/sottoscrizione locale per comunicare su un dispositivo principale. Per consentire a un componente personalizzato di reagire ai cambiamenti dello stato ombra, puoi iscriverti agli argomenti di pubblicazione/sottoscrizione locali. Ciò consente al componente di ricevere messaggi sugli argomenti shadow locali e quindi di agire su tali messaggi.

Gli argomenti shadow locali utilizzano lo stesso formato degli argomenti MQTT di AWS IoT Device Shadow. Per ulteriori informazioni sugli argomenti shadow, consultate gli argomenti Device Shadow MQTT nella AWS IoTDevice Shadow Guide.

Per reagire ai cambiamenti dello stato ombra locale

1. Aggiungi le politiche di controllo degli accessi alla ricetta del tuo componente personalizzato per consentire al componente di ricevere messaggi su argomenti shadow locali.

   Ad esempio, le politiche di autorizzazione, vedi Esempi di politiche di autorizzazione IPC Shadow Local.
2. Per avviare un'azione personalizzata in un componente, utilizzate le operazioni `SubscribeToTopic` IPC per sottoscrivere gli argomenti shadow su cui desiderate ricevere messaggi. Per ulteriori informazioni sull'utilizzo delle operazioni IPC di pubblicazione/sottoscrizione locali nel codice del componente, vedere. Pubblicare/sottoscrivere messaggi locali
3. Per richiamare una funzione Lambda, usa la configurazione dell'origine dell'evento per fornire il nome dell'argomento shadow e specifica che si tratta di un argomento di pubblicazione/sottoscrizione locale. Per informazioni sulla creazione di componenti della funzione Lambda, vedere. Esegui AWS Lambda funzioni

> **Note**
>
> Per consentire a un dispositivo principale di interagire con le ombre dei dispositivi client, è inoltre necessario configurare e distribuire il componente bridge MQTT. Per ulteriori informazioni, consultate Attivare lo shadow manager per comunicare con i dispositivi client.

Sincronizza le ombre del dispositivo locale con AWS IoT Core

Il componente shadow manager consente di AWS IoT Greengrass sincronizzare gli stati ombra del dispositivo locale con. AWS IoT Core È necessario modificare la configurazione del componente shadow manager per includere il parametro di `synchronization` configurazione e specificare i nomi AWS IoT degli oggetti per i dispositivi e le ombre che si desidera sincronizzare.

Quando configurate Shadow Manager per sincronizzare le ombre, sincronizza tutte le modifiche di stato per le ombre specificate, indipendentemente dal fatto che le modifiche avvengano nei documenti shadow locali o nei documenti shadow nel cloud.

È inoltre possibile specificare se il componente shadow manager sincronizza le ombre in tempo reale o a intervalli periodici. Per impostazione predefinita, il componente shadow manager sincronizza le ombre in tempo reale, in modo che il dispositivo principale invii e riceva gli aggiornamenti delle ombre da e verso ogni aggiornamento. AWS IoT Core È possibile configurare intervalli periodici per ridurre l'utilizzo della larghezza di banda e i costi.

Argomenti

Prerequisiti

Per sincronizzare le ombre locali conAWS IoT Core, è necessario configurare la politica del dispositivo principale AWS IoT di Greengrass per consentire le AWS IoT Core seguenti azioni di policy ombra.

- `iot:GetThingShadow`

- `iot:UpdateThingShadow`
- `iot:DeleteThingShadow`

Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS IoT Coreazioni politiche nella Guida per gli sviluppatori AWS IoT
- AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali
- Aggiorna la politica di un dispositivo principale AWS IoT

Configurare il componente shadow manager

Lo shadow manager richiede un elenco di mappature dei nomi shadow per sincronizzare le informazioni sullo stato ombra nei documenti shadow locali con i documenti shadow nel cloud. AWS IoT Core

Per sincronizzare gli stati shadow, create una distribuzione che includa il `aws.greengrass.ShadowManager` componente e specificate le shadow che desiderate sincronizzare nel parametro di `synchronize` configurazione nella configurazione dello shadow manager nella distribuzione.

> **Note**
>
> Per consentire a un dispositivo principale di interagire con le ombre dei dispositivi client, è inoltre necessario configurare e distribuire il componente bridge MQTT. Per ulteriori informazioni, consultate Attivare lo shadow manager per comunicare con i dispositivi client.

L'aggiornamento della configurazione di esempio seguente indica al componente shadow manager di sincronizzare le seguenti ombre con: AWS IoT Core

- L'ombra classica per il dispositivo principale
- Il nome `MyCoreShadow` del dispositivo principale
- La classica ombra per una cosa chiamata IoT `MyDevice2`
- Le ombre denominate `MyShadowA` e `MyShadowB` per una cosa chiamata IoT `MyDevice1`

Questo aggiornamento della configurazione specifica di sincronizzare le ombre con AWS IoT Core in tempo reale. Se si utilizza shadow manager v2.1.0 o versione successiva, è possibile configurare il componente shadow manager per sincronizzare le ombre a intervalli periodici. Per configurare questa funzionalità, modificate la strategia di sincronizzazione in e specificate un intervallo `delay` in secondi. `periodic` Per ulteriori informazioni, vedete il parametro di configurazione della strategia del componente shadow manager.

Questo aggiornamento della configurazione specifica di sincronizzare le ombre in entrambe le direzioni tra AWS IoT Core e il dispositivo principale. Se si utilizza shadow manager v2.2.0 o versione successiva, è possibile configurare il componente Shadow Manager per sincronizzare le ombre in una sola direzione. Per configurare questa funzionalità, modifica la sincronizzazione `direction` in o. `deviceToCloud` `cloudToDevice` Per ulteriori informazioni, vedete il parametro di configurazione della direzione del componente shadow manager.

```
{
  "strategy": {
    "type": "realTime"
  },
  "synchronize": {
    "coreThing": {
      "classic": true,
      "namedShadows": [
        "MyCoreShadow"
      ]
    },
    "shadowDocuments": [
      {
        "thingName": "MyDevice1",
        "classic": false,
        "namedShadows": [
          "MyShadowA",
          "MyShadowB"
        ]
      },
      {
        "thingName": "MyDevice2",
        "classic": true,
        "namedShadows": [ ]
      }
    ],
    "direction": "betweenDeviceAndCloud"
  }
```

```
}
```

Sincronizza le ombre locali

Quando il dispositivo principale Greengrass è connesso al AWS IoT cloud, lo shadow manager esegue le seguenti attività per le ombre specificate nella configurazione del componente. Il comportamento dipende dall'opzione di configurazione della direzione di sincronizzazione delle ombre specificata. Per impostazione predefinita, Shadow Manager utilizza l'`betweenDeviceAndCloud`opzione per sincronizzare le ombre in entrambe le direzioni. Se si utilizza shadow manager v2.2.0 o versione successiva, è possibile configurare il dispositivo principale per sincronizzare le ombre in una sola direzione, che può essere o. `cloudToDevice` `deviceToCloud`

- Se la configurazione della direzione di sincronizzazione dello shadow è `betweenDeviceAndCloud` o`cloudToDevice`, shadow manager recupera le informazioni sullo stato riportate dal documento cloud shadow in. AWS IoT Core Quindi, aggiorna i documenti shadow archiviati localmente per sincronizzare lo stato del dispositivo.
- Se la configurazione della direzione di sincronizzazione dello shadow è `betweenDeviceAndCloud` o`deviceToCloud`, shadow manager pubblica lo stato corrente del dispositivo nel documento cloud shadow.

Comportamento dei conflitti di fusione delle ombre

In alcuni casi, ad esempio quando il dispositivo principale è disconnesso da Internet, uno shadow potrebbe cambiare nel servizio shadow locale e nel AWS IoT cloud prima che lo shadow manager sincronizzi le modifiche. Di conseguenza, gli stati desiderati e quelli riportati differiscono tra il servizio shadow locale e il cloud AWS IoT

Quando lo shadow manager sincronizza l'ombra, unisce le modifiche in base al seguente comportamento:

- Se utilizzate una versione di shadow manager precedente alla v2.2.0 o quando specificate la direzione di sincronizzazione dello `betweenDeviceAndCloud` shadow, si applica il seguente comportamento:
  - Quando si verifica un conflitto di fusione nello stato desiderato di un'ombra, lo shadow manager sovrascrive la sezione in conflitto del documento shadow locale con il valore proveniente dal cloud. AWS IoT

  - Quando si verifica un conflitto di unione nello stato riportato da un'ombra, lo shadow manager sovrascrive la sezione in conflitto dell'ombra nel AWS IoT cloud con il valore del documento shadow locale.
- Quando specificate la `deviceToCloud` direzione di sincronizzazione delle ombre, lo shadow manager sovrascrive la sezione in conflitto dell'ombra nel AWS IoT cloud con il valore del documento shadow locale.
- Quando specificate la direzione di sincronizzazione dello `cloudToDevice` shadow, lo shadow manager sovrascrive la sezione in conflitto del documento shadow locale con il valore proveniente dal cloud. AWS IoT

Gestisci i flussi di dati sui dispositivi core Greengrass

AWS IoT Greengrassstream manager rende più efficiente e affidabile il trasferimento di dati IoT ad alto volume a. Cloud AWS Stream manager elabora i flussi di dati sul AWS IoT Greengrass Core prima di esportarli in. Cloud AWS Stream Manager si integra con scenari edge comuni, come l'inferenza di machine learning (ML), in cui il dispositivo AWS IoT Greengrass Core elabora e analizza i dati prima di esportarli verso le destinazioni di archiviazione Cloud AWS o locali.

Stream Manager fornisce un'interfaccia comune per semplificare lo sviluppo di componenti personalizzati in modo da non dover creare funzionalità di gestione dei flussi personalizzate. I componenti possono utilizzare un meccanismo standardizzato per elaborare flussi ad alto volume e gestire le politiche locali di conservazione dei dati. È possibile definire politiche per il tipo di archiviazione, le dimensioni e la conservazione dei dati per ogni flusso per controllare il modo in cui lo stream manager elabora ed esporta i dati.

Stream Manager funziona in ambienti con connettività intermittente o limitata. È possibile definire l'utilizzo della larghezza di banda, il comportamento di timeout e il modo in cui AWS IoT Greengrass Core gestisce i dati di streaming quando è connesso o disconnesso. Puoi anche impostare priorità per controllare l'ordine in cui AWS IoT Greengrass Core esporta i flussi verso. Cloud AWS In questo modo è possibile gestire i dati critici prima degli altri dati.

È possibile configurare lo stream manager per esportare automaticamente i dati verso l'Cloud AWSarchiviazione o l'ulteriore elaborazione e analisi. Stream manager supporta le esportazioni verso le seguenti Cloud AWS destinazioni:

- Canali inAWS IoT Analytics. AWS IoT Analyticsconsente di eseguire analisi avanzate sui dati per aiutare a prendere decisioni aziendali e migliorare i modelli di apprendimento automatico. Per ulteriori informazioni, consulta Che cos'è AWS IoT Analytics? nella Guida per l'utente di AWS IoT Analytics.
- Stream in Amazon Kinesis Data Streams. Puoi utilizzare Kinesis Data Streams per aggregare dati di grandi volumi e caricarli in un data warehouse o cluster. MapReduce Per ulteriori informazioni, consulta Cos'è Amazon Kinesis Data Streams? nella Guida per gli sviluppatori di Amazon Kinesis Data Streams.
- Proprietà delle risorse in. AWS IoT SiteWise AWS IoT SiteWiseconsente di raccogliere, organizzare e analizzare i dati provenienti da apparecchiature industriali su larga scala. Per ulteriori informazioni, consulta Che cos'è AWS IoT SiteWise? nella Guida per l'utente di AWS IoT SiteWise.

- Oggetti in Amazon Simple Storage Service Amazon S3. Puoi usare Amazon S3 per archiviare e recuperare grandi quantità di dati. Per ulteriori informazioni, consulta Cos'è Amazon S3? nella Guida per sviluppatori di Amazon Simple Storage Service.

Flusso di lavoro della gestione dei flussi

Le tue applicazioni IoT interagiscono con Stream Manager tramite Stream Manager SDK.

In un flusso di lavoro semplice, un componente AWS IoT Greengrass centrale consuma dati IoT, come le metriche di temperatura e pressione in serie temporali. Il componente potrebbe filtrare o comprimere i dati e quindi chiamare l'SDK Stream Manager per scrivere i dati in uno stream in stream manager. Stream Manager può esportare lo stream in Cloud AWS modo automatico in base alle politiche definite per lo stream. I componenti possono anche inviare dati direttamente ai database o agli archivi di archiviazione locali.

Le tue applicazioni IoT possono includere più componenti personalizzati che leggono o scrivono negli stream. Questi componenti possono leggere e scrivere negli stream per filtrare, aggregare e analizzare i dati sul AWS IoT Greengrass dispositivo principale. In questo modo è possibile rispondere rapidamente agli eventi locali ed estrarre informazioni preziose prima che i dati vengano trasferiti dalle destinazioni principali a quelle Cloud AWS locali.

Per iniziare, implementa il componente stream manager sul tuo dispositivo AWS IoT Greengrass principale. Nella distribuzione, configura i parametri del componente stream manager per definire le impostazioni che si applicano a tutti gli stream sul dispositivo principale Greengrass. Utilizzate questi parametri per controllare il modo in cui lo stream manager archivia, elabora ed esporta i flussi in base alle esigenze aziendali e ai vincoli ambientali.

Dopo aver configurato lo stream manager, puoi creare e distribuire le tue applicazioni IoT. Si tratta in genere di componenti personalizzati che vengono utilizzati `StreamManagerClient` nell'SDK Stream Manager per creare e interagire con gli stream. Quando crei uno stream, puoi definire politiche per flusso, come destinazioni di esportazione, priorità e persistenza.

Requisiti

Per l'utilizzo di stream manager si applicano i seguenti requisiti:

- Stream manager richiede un minimo di 70 MB di RAM oltre al software AWS IoT Greengrass Core. Il requisito di memoria totale dipende dal carico di lavoro.

- AWS IoT Greengrassi componenti devono utilizzare lo Stream Manager SDK per interagire con lo stream manager. L'SDK Stream Manager è disponibile nelle seguenti lingue:
  - Stream Manager SDK per Java (v1.1.0 o versione successiva)
  - Stream Manager SDK per Node.js (v1.1.0 o versione successiva)
  - Stream Manager SDK per Python (v1.1.0 o successivo)
- AWS IoT Greengrassi componenti devono specificare lo stream manager component (`aws.greengrass.StreamManager`) come dipendenza nella loro ricetta per utilizzare stream manager.

  > **Note**
  >
  > Se utilizzi stream manager per esportare dati nel cloud, non puoi aggiornare la versione 2.0.7 del componente stream manager a una versione compresa tra v2.0.8 e v2.0.11. Se stai implementando stream manager per la prima volta, ti consigliamo vivamente di distribuire la versione più recente del componente stream manager.

- Se definisci le destinazioni di Cloud AWS esportazione per uno stream, devi creare i tuoi obiettivi di esportazione e concedere le autorizzazioni di accesso nel ruolo del dispositivo Greengrass. A seconda della destinazione, potrebbero applicarsi anche altri requisiti. Per ulteriori informazioni, consultare:
  - the section called “Canali AWS IoT Analytics”
  - the section called “Flussi di dati Amazon Kinesis”
  - the section called “AWS IoT SiteWiseproprietà degli asset”
  - the section called “Oggetti Amazon S3”

  L'utente è responsabile del mantenimento di queste Cloud AWS risorse.

Sicurezza dei dati

Quando utilizzi stream manager, tiene presente le seguenti considerazioni di sicurezza.

Sicurezza dei dati locali

AWS IoT Greengrassnon crittografa i dati di flusso inattivi o in transito tra i componenti locali del dispositivo principale.

- Dati inattivi. I dati del flusso vengono archiviati localmente in una directory di storage. Per la sicurezza dei dati, AWS IoT Greengrass si affida alle autorizzazioni relative ai file e alla crittografia dell'intero disco, se abilitata. Puoi utilizzare il parametro STREAM_MANAGER_STORE_ROOT_DIR opzionale per specificare la directory di storage. Se modifichi questo parametro in un secondo momento per utilizzare una directory di storage diversa, AWS IoT Greengrass non elimina la directory di storage precedente o il relativo contenuto.
- Dati in transito a livello locale. AWS IoT Greengrassnon crittografa i dati di flusso in transito locale tra fonti di dati, AWS IoT Greengrass componenti, Stream Manager SDK e stream manager.
- Dati in transito verso. Cloud AWS I flussi di dati esportati da Stream Manager Cloud AWS utilizzano la crittografia client di AWS servizio standard con Transport Layer Security (TLS).

Autenticazione client

I client Stream Manager utilizzano Stream Manager SDK per comunicare con lo stream manager. Quando l'autenticazione client è abilitata, solo i componenti Greengrass possono interagire con gli stream in stream manager. Quando l'autenticazione del client è disabilitata, qualsiasi processo in esecuzione sul dispositivo principale Greengrass può interagire con gli stream in stream manager. È opportuno disabilitare l'autenticazione solo se richiesto dal business case.

Utilizza il parametro STREAM_MANAGER_AUTHENTICATE_CLIENT per impostare la modalità di autenticazione client. È possibile configurare questo parametro quando si distribuisce il componente stream manager sui dispositivi principali.

	Abilitato	Disabilitato
Valore del parametro	`true` (predefinito e consiglia to)	`false`
Client consentiti	Componenti Greengrass sul dispositivo principale	Componenti Greengrass sul dispositivo principale Altri processi in esecuzione sul dispositivo core Greengrass

Consulta anche

- the section called “Configurazione di Stream Manager”
- the section called “Utilizzalo StreamManagerClient per lavorare con gli stream”
- the section called “Esporta le configurazioni per le destinazioni cloud supportate”

Crea componenti personalizzati che utilizzano stream manager

Usa stream manager nei componenti Greengrass personalizzati per archiviare, elaborare ed esportare i dati dei dispositivi IoT. Usa le procedure e gli esempi in questa sezione per creare ricette di componenti, artefatti e applicazioni che funzionano con stream manager. Per ulteriori informazioni su come sviluppare e testare i componenti, consultareCrea AWS IoT Greengrass componenti.

Argomenti

- Definisci le ricette dei componenti che utilizzano stream manager
- Connect allo stream manager nel codice dell'applicazione

Definisci le ricette dei componenti che utilizzano stream manager

Per utilizzare stream manager in un componente personalizzato, è necessario definire il `aws.greengrass.StreamManager` componente come dipendenza. È inoltre necessario fornire lo Stream Manager SDK. Completa le seguenti attività per scaricare e utilizzare Stream Manager SDK nella lingua di tua scelta.

Usa lo Stream Manager SDK for Java

Lo Stream Manager SDK for Java è disponibile come file JAR che puoi usare per compilare il tuo componente. Quindi, puoi creare un JAR dell'applicazione che includa Stream Manager SDK, definire il JAR dell'applicazione come elemento del componente ed eseguire il JAR dell'applicazione nel ciclo di vita del componente.

Per utilizzare Stream Manager SDK for Java

1. Scarica il file JAR di Stream Manager SDK for Java.
2. Effettuate una delle seguenti operazioni per creare gli artefatti dei componenti dall'applicazione Java e dal file JAR SDK di Stream Manager:

- Crea la tua applicazione come file JAR che include Stream Manager SDK JAR ed esegui questo file JAR nella ricetta del componente.
- Definisci lo Stream Manager SDK JAR come elemento componente. Aggiungi quell'artefatto al classpath quando esegui l'applicazione nella ricetta del componente.

La ricetta dei componenti potrebbe essere simile al seguente esempio. Questo componente esegue una versione modificata dell'esempio StreamManagerS3.java, in cui `StreamManagerS3.jar` include Stream Manager SDK JAR.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.StreamManagerS3Java",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Uses stream manager to upload a file to an S3
 bucket.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.StreamManager": {
      "VersionRequirement": "^2.0.0"
    }
  },
  "Manifests": [
    {
      "Lifecycle": {
        "run": "java -jar {artifacts:path}/StreamManagerS3.jar"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Java/1.0.0/StreamManagerS3.jar"
        }
      ]
    }
  ]
}
```

YAML

```
---
```

```
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.StreamManagerS3Java
ComponentVersion: 1.0.0
ComponentDescription: Uses stream manager to upload a file to an S3 bucket.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.StreamManager:
    VersionRequirement: "^2.0.0"
Manifests:
  - Lifecycle:
      run: java -jar {artifacts:path}/StreamManagerS3.jar
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Java/1.0.0/StreamManagerS3.jar
```

Per ulteriori informazioni su come sviluppare e testare i componenti, consultareCrea AWS IoT Greengrass componenti.

Usa lo Stream Manager SDK per Python

Lo Stream Manager SDK per Python è disponibile come codice sorgente che puoi includere nel tuo componente. Crea un file ZIP di Stream Manager SDK, definisci il file ZIP come elemento del componente e installa i requisiti dell'SDK nel ciclo di vita del componente.

Per usare lo Stream Manager SDK per Python

1. Clona o scarica il repository aws-greengrass-stream-manager-sdk-python.

   ```
   git clone git@github.com:aws-greengrass/aws-greengrass-stream-manager-sdk-
   python.git
   ```

2. Crea un file ZIP che contenga la `stream_manager` cartella, che contiene il codice sorgente dello Stream Manager SDK per Python. Puoi fornire questo file ZIP come elemento componente che il software AWS IoT Greengrass Core decomprime quando installa il componente. Esegui questa operazione:

   a. Aprire la cartella contenente il repository scaricato o clonato nel passaggio precedente.

      ```
      cd aws-greengrass-stream-manager-sdk-python
      ```

b. Comprimi la `stream_manager` cartella in un file ZIP denominato`stream_manager_sdk.zip`.

Linux or Unix

```
zip -rv stream_manager_sdk.zip stream_manager
```

Windows Command Prompt (CMD)

```
tar -acvf stream_manager_sdk.zip stream_manager
```

PowerShell

```
Compress-Archive stream_manager stream_manager_sdk.zip
```

c. Verifica che il `stream_manager_sdk.zip` file contenga la `stream_manager` cartella e il suo contenuto. Esegui il comando seguente per elencare il contenuto del file ZIP.

Linux or Unix

```
unzip -l stream_manager_sdk.zip
```

Windows Command Prompt (CMD)

```
tar -tf stream_manager_sdk.zip
```

L'output visualizzato dovrebbe essere simile al seguente:

```
Archive:  aws-greengrass-stream-manager-sdk-python/stream_manager.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  02-24-2021 20:45   stream_manager/
      913  02-24-2021 20:45   stream_manager/__init__.py
     9719  02-24-2021 20:45   stream_manager/utilinternal.py
     1412  02-24-2021 20:45   stream_manager/exceptions.py
     1004  02-24-2021 20:45   stream_manager/util.py
        0  02-24-2021 20:45   stream_manager/data/
   254463  02-24-2021 20:45   stream_manager/data/__init__.py
    26515  02-24-2021 20:45   stream_manager/streammanagerclient.py
```

```
---------                    -------
   294026                    8 files
```

3. Copia gli artefatti dell'SDK di Stream Manager nella cartella degli artefatti del tuo componente. Oltre al file ZIP di Stream Manager SDK, il componente utilizza il file dell'SDK per installare le dipendenze dello Stream Manager SDK. `requirements.txt` Sostituisci *`~/greengrass-components`* con il percorso della cartella che usi per lo sviluppo locale.

   Linux or Unix

   ```
   cp {stream_manager_sdk.zip,requirements.txt} ~/greengrass-components/artifacts/
   com.example.StreamManagerS3Python/1.0.0/
   ```

   Windows Command Prompt (CMD)

   ```
   robocopy . %USERPROFILE%\greengrass-components\artifacts
   \com.example.StreamManagerS3Python\1.0.0 stream_manager_sdk.zip
   robocopy . %USERPROFILE%\greengrass-components\artifacts
   \com.example.StreamManagerS3Python\1.0.0 requirements.txt
   ```

   PowerShell

   ```
   cp .\stream_manager_sdk.zip,.\requirements.txt ~\greengrass-components\artifacts
   \com.example.StreamManagerS3Python\1.0.0\
   ```

4. Crea la ricetta dei tuoi componenti. Nella ricetta, esegui queste operazioni:

   a. Definisci `stream_manager_sdk.zip` e `requirements.txt` come artefatti.

   b. Definisci la tua applicazione Python come un artefatto.

   c. Nel ciclo di vita dell'installazione, installa i requisiti SDK di Stream Manager da. `requirements.txt`

   d. Nel ciclo di vita dell'esecuzione, aggiungi Stream Manager SDK ed esegui la tua applicazione PYTHONPATH Python.

   La ricetta dei componenti potrebbe essere simile al seguente esempio. Questo componente esegue l'esempio stream_manager_s3.py.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.StreamManagerS3Python",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Uses stream manager to upload a file to an S3
 bucket.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.StreamManager": {
      "VersionRequirement": "^2.0.0"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "pip3 install --user -r {artifacts:path}/requirements.txt",
        "run": "export PYTHONPATH=$PYTHONPATH:{artifacts:decompressedPath}/
stream_manager_sdk; python3 {artifacts:path}/stream_manager_s3.py"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_sdk.zip",
          "Unarchive": "ZIP"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_s3.py"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/requirements.txt"
        }
      ]
    },
    {
      "Platform": {
        "os": "windows"
```

```
      },
      "Lifecycle": {
        "install": "pip3 install --user -r {artifacts:path}/requirements.txt",
        "run": "set \"PYTHONPATH=%PYTHONPATH%;{artifacts:decompressedPath}/
stream_manager_sdk\" & py -3 {artifacts:path}/stream_manager_s3.py"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_sdk.zip",
          "Unarchive": "ZIP"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_s3.py"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/requirements.txt"
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.StreamManagerS3Python
ComponentVersion: 1.0.0
ComponentDescription: Uses stream manager to upload a file to an S3 bucket.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.StreamManager:
    VersionRequirement: "^2.0.0"
Manifests:
  - Platform:
      os: linux
    Lifecycle:
      install: pip3 install --user -r {artifacts:path}/requirements.txt
      run: |
```

```
        export PYTHONPATH=$PYTHONPATH:{artifacts:decompressedPath}/
stream_manager_sdk
        python3 {artifacts:path}/stream_manager_s3.py
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_sdk.zip
        Unarchive: ZIP
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_s3.py
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/requirements.txt
  - Platform:
      os: windows
    Lifecycle:
      install: pip3 install --user -r {artifacts:path}/requirements.txt
      run: |
        set "PYTHONPATH=%PYTHONPATH%;{artifacts:decompressedPath}/
stream_manager_sdk"
        py -3 {artifacts:path}/stream_manager_s3.py
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_sdk.zip
        Unarchive: ZIP
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/stream_manager_s3.py
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3Python/1.0.0/requirements.txt
```

Per ulteriori informazioni su come sviluppare e testare i componenti, consultareCrea AWS IoT Greengrass componenti.

Usa Stream Manager SDK per JavaScript

Lo Stream Manager SDK per JavaScript è disponibile come codice sorgente che puoi includere nel tuo componente. Crea un file ZIP di Stream Manager SDK, definisci il file ZIP come elemento del componente e installa l'SDK nel ciclo di vita del componente.

Per utilizzare Stream Manager SDK per JavaScript

1. Clona o scarica il repository aws-greengrass-stream-manager-sdk-js.

```
git clone git@github.com:aws-greengrass/aws-greengrass-stream-manager-sdk-js.git
```

2. Crea un file ZIP che contenga la `aws-greengrass-stream-manager-sdk` cartella, che contiene il codice sorgente di Stream Manager SDK perJavaScript. Puoi fornire questo file ZIP come elemento componente che il software AWS IoT Greengrass Core decomprime quando installa il componente. Esegui questa operazione:

   a. Aprire la cartella contenente il repository scaricato o clonato nel passaggio precedente.

   ```
   cd aws-greengrass-stream-manager-sdk-js
   ```

   b. Comprimi la `aws-greengrass-stream-manager-sdk` cartella in un file ZIP denominato`stream-manager-sdk.zip`.

   Linux or Unix

   ```
   zip -rv stream-manager-sdk.zip aws-greengrass-stream-manager-sdk
   ```

   Windows Command Prompt (CMD)

   ```
   tar -acvf stream-manager-sdk.zip aws-greengrass-stream-manager-sdk
   ```

   PowerShell

   ```
   Compress-Archive aws-greengrass-stream-manager-sdk stream-manager-sdk.zip
   ```

   c. Verifica che il `stream-manager-sdk.zip` file contenga la `aws-greengrass-stream-manager-sdk` cartella e il suo contenuto. Esegui il comando seguente per elencare il contenuto del file ZIP.

   Linux or Unix

   ```
   unzip -l stream-manager-sdk.zip
   ```

   Windows Command Prompt (CMD)

   ```
   tar -tf stream-manager-sdk.zip
   ```

L'output visualizzato dovrebbe essere simile al seguente:

```
Archive:  stream-manager-sdk.zip
  Length      Date    Time    Name
---------  ---------- -----   ----
        0  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/
      369  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/package.json
     1017  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/util.js
     8374  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/utilInternal.js
     1937  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/exceptions.js
        0  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/data/
   353343  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/data/index.js
    22599  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/client.js
      216  02-24-2021 22:36   aws-greengrass-stream-manager-sdk/index.js
---------                     -------
   387855                     9 files
```

3. Copia l'elemento SDK di Stream Manager nella cartella degli artefatti del tuo componente. Sostituisci *~/greengrass-components* con il percorso della cartella che usi per lo sviluppo locale.

   Linux or Unix

   ```
   cp stream-manager-sdk.zip ~/greengrass-components/artifacts/
   com.example.StreamManagerS3JS/1.0.0/
   ```

   Windows Command Prompt (CMD)

   ```
   robocopy . %USERPROFILE%\greengrass-components\artifacts
   \com.example.StreamManagerS3JS\1.0.0 stream-manager-sdk.zip
   ```

   PowerShell

   ```
   cp .\stream-manager-sdk.zip ~\greengrass-components\artifacts
   \com.example.StreamManagerS3JS\1.0.0\
   ```

4. Crea la ricetta dei tuoi componenti. Nella ricetta, esegui queste operazioni:

   a. `stream-manager-sdk.zip`Definiscilo come un artefatto.

   b. Definisci la tua JavaScript applicazione come un artefatto.

c. Nel ciclo di vita dell'installazione, installa Stream Manager SDK dall'artefatto. `stream-manager-sdk.zip` Questo `npm install` comando crea una `node_modules` cartella che contiene lo Stream Manager SDK e le sue dipendenze.

d. Nel ciclo di vita dell'esecuzione, aggiungi la `node_modules` cartella ed esegui `NODE_PATH` l'applicazione. JavaScript

La ricetta dei componenti potrebbe essere simile al seguente esempio. Questo componente esegue l'esempio StreamManagerS3.

JSON

```
{
  "RecipeFormatVersion": "2020-01-25",
  "ComponentName": "com.example.StreamManagerS3JS",
  "ComponentVersion": "1.0.0",
  "ComponentDescription": "Uses stream manager to upload a file to an S3
 bucket.",
  "ComponentPublisher": "Amazon",
  "ComponentDependencies": {
    "aws.greengrass.StreamManager": {
      "VersionRequirement": "^2.0.0"
    }
  },
  "Manifests": [
    {
      "Platform": {
        "os": "linux"
      },
      "Lifecycle": {
        "install": "npm install {artifacts:decompressedPath}/stream-manager-sdk/
aws-greengrass-stream-manager-sdk",
        "run": "export NODE_PATH=$NODE_PATH:{work:path}/node_modules; node
 {artifacts:path}/index.js"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/stream-manager-sdk.zip",
          "Unarchive": "ZIP"
        },
        {
```

```
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/index.js"
        }
      ]
    },
    {
      "Platform": {
        "os": "windows"
      },
      "Lifecycle": {
        "install": "npm install {artifacts:decompressedPath}/stream-manager-sdk/
aws-greengrass-stream-manager-sdk",
        "run": "set \"NODE_PATH=%NODE_PATH%;{work:path}/node_modules\" & node
 {artifacts:path}/index.js"
      },
      "Artifacts": [
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/stream-manager-sdk.zip",
          "Unarchive": "ZIP"
        },
        {
          "URI": "s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/index.js"
        }
      ]
    }
  ]
}
```

YAML

```
---
RecipeFormatVersion: '2020-01-25'
ComponentName: com.example.StreamManagerS3JS
ComponentVersion: 1.0.0
ComponentDescription: Uses stream manager to upload a file to an S3 bucket.
ComponentPublisher: Amazon
ComponentDependencies:
  aws.greengrass.StreamManager:
    VersionRequirement: "^2.0.0"
Manifests:
  - Platform:
```

```
      os: linux
    Lifecycle:
      install: npm install {artifacts:decompressedPath}/stream-manager-sdk/aws-
greengrass-stream-manager-sdk
      run: |
        export NODE_PATH=$NODE_PATH:{work:path}/node_modules
        node {artifacts:path}/index.js
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/stream-manager-sdk.zip
        Unarchive: ZIP
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/index.js
  - Platform:
      os: windows
    Lifecycle:
      install: npm install {artifacts:decompressedPath}/stream-manager-sdk/aws-
greengrass-stream-manager-sdk
      run: |
        set "NODE_PATH=%NODE_PATH%;{work:path}/node_modules"
        node {artifacts:path}/index.js
    Artifacts:
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/stream-manager-sdk.zip
        Unarchive: ZIP
      - URI: s3://DOC-EXAMPLE-BUCKET/artifacts/
com.example.StreamManagerS3JS/1.0.0/index.js
```

Per ulteriori informazioni su come sviluppare e testare i componenti, consultareCrea AWS IoT Greengrass componenti.

Connect allo stream manager nel codice dell'applicazione

Per connetterti allo stream manager nella tua applicazione, crea un'istanza di `StreamManagerClient` da Stream Manager SDK. Questo client si connette al componente Stream Manager sulla porta predefinita 8088 o sulla porta specificata. Per ulteriori informazioni su come utilizzare `StreamManagerClient` dopo aver creato un'istanza, consultaUtilizzalo StreamManagerClient per lavorare con gli stream.

Example Esempio: Connect allo stream manager con la porta predefinita

Java

```
import com.amazonaws.greengrass.streammanager.client.StreamManagerClient;

public class MyStreamManagerComponent {

    void connectToStreamManagerWithDefaultPort() {
        StreamManagerClient client = StreamManagerClientFactory.standard().build();

        // Use the client.
    }
}
```

Python

```
from stream_manager import (
    StreamManagerClient
)

def connect_to_stream_manager_with_default_port():
    client = StreamManagerClient()

    # Use the client.
```

JavaScript

```
const {
    StreamManagerClient
} = require('aws-greengrass-stream-manager-sdk');

function connectToStreamManagerWithDefaultPort() {
    const client = new StreamManagerClient();

    // Use the client.
}
```

Example Esempio: Connect a stream manager con una porta non predefinita

Se configuri stream manager con una porta diversa da quella predefinita, devi utilizzare la comunicazione tra processi per recuperare la porta dalla configurazione del componente.

> **Note**
>
> Il parametro di `port` configurazione contiene il valore specificato `STREAM_MANAGER_SERVER_PORT` quando si distribuisce stream manager.

Java

```
void connectToStreamManagerWithCustomPort() {
    EventStreamRPCConnection eventStreamRpcConnection =
 IPCUtils.getEventStreamRpcConnection();
    GreengrassCoreIPCClient greengrassCoreIPCClient = new
 GreengrassCoreIPCClient(eventStreamRpcConnection);
    List<String> keyPath = new ArrayList<>();
    keyPath.add("port");

    GetConfigurationRequest request = new GetConfigurationRequest();
    request.setComponentName("aws.greengrass.StreamManager");
    request.setKeyPath(keyPath);
    GetConfigurationResponse response =
            greengrassCoreIPCClient.getConfiguration(request,
 Optional.empty()).getResponse().get();
    String port = response.getValue().get("port").toString();
    System.out.print("Stream Manager is running on port: " + port);

    final StreamManagerClientConfig config = StreamManagerClientConfig.builder()

 .serverInfo(StreamManagerServerInfo.builder().port(Integer.parseInt(port)).build()).build()

    StreamManagerClient client =
 StreamManagerClientFactory.standard().withClientConfig(config).build();

    // Use the client.
}
```

Python

```
import awsiot.greengrasscoreipc
from awsiot.greengrasscoreipc.model import (
    GetConfigurationRequest
)
from stream_manager import (
    StreamManagerClient
)

TIMEOUT = 10

def connect_to_stream_manager_with_custom_port():
    # Use IPC to get the port from the stream manager component configuration.
    ipc_client = awsiot.greengrasscoreipc.connect()
    request = GetConfigurationRequest()
    request.component_name = "aws.greengrass.StreamManager"
    request.key_path = ["port"]
    operation = ipc_client.new_get_configuration()
    operation.activate(request)
    future_response = operation.get_response()
    response = future_response.result(TIMEOUT)
    stream_manager_port = str(response.value["port"])

    # Use port to create a stream manager client.
    stream_client = StreamManagerClient(port=stream_manager_port)

    # Use the client.
```

Utilizzalo StreamManagerClient per lavorare con gli stream

I componenti Greengrass definiti dall'utente che vengono eseguiti sul dispositivo principale Greengrass possono utilizzare `StreamManagerClient` l'oggetto nell'SDK Stream Manager per creare flussi in stream manager e quindi interagire con i flussi. Quando un componente crea uno stream, definisce le Cloud AWS destinazioni, la prioritizzazione e altre politiche di esportazione e conservazione dei dati per lo stream. Per inviare dati allo stream manager, i componenti aggiungono i dati allo stream. Se viene definita una destinazione di esportazione per lo stream, lo stream manager esporta lo stream automaticamente.

> **Note**
>
> In genere, i client di stream manager sono componenti Greengrass definiti dall'utente. Se il tuo business case lo richiede, puoi anche consentire ai processi non componenti in esecuzione sul core di Greengrass (ad esempio, un contenitore Docker) di interagire con lo stream manager. Per ulteriori informazioni, consulta the section called “Autenticazione client”.

Gli snippet di questo argomento mostrano come i client chiamano i `StreamManagerClient` metodi per lavorare con gli stream. Per i dettagli di implementazione sui metodi e i relativi argomenti, utilizzate i collegamenti al riferimento SDK elencati dopo ogni frammento.

Se usi stream manager in una funzione Lambda, la tua funzione Lambda dovrebbe creare un'istanza `StreamManagerClient` all'esterno del gestore della funzione. Se viene creata un'istanza nel gestore, la funzione crea un `client` e una connessione al gestore flussi ogni volta che viene richiamata.

> **Note**
>
> Se si esegue un'istanza `StreamManagerClient` nel gestore, è necessario chiamare esplicitamente il metodo `close()` quando `client` completa il suo lavoro. In caso contrario, `client` mantiene la connessione aperta e un altro thread in esecuzione fino alla chiusura dello script.

`StreamManagerClient` supporta le seguenti operazioni:

- the section called “Creazione del flusso di messaggi”
- the section called “Aggiunta di un messaggio”
- the section called “Lettura di messaggi”
- the section called “Visualizzazione dell'elenco di flussi”
- the section called “Descrizione del flusso di messaggi”
- the section called “Aggiorna il flusso di messaggi”
- the section called “Eliminazione del flusso di messaggi”

Creazione del flusso di messaggi

Per creare uno stream, un componente Greengrass definito dall'utente chiama il metodo create e passa un oggetto. `MessageStreamDefinition` Questo oggetto specifica il nome univoco dello stream e definisce come lo stream manager deve gestire i nuovi dati quando viene raggiunta la dimensione massima del flusso. È possibile utilizzare `MessageStreamDefinition` e relativi tipi di dati (ad esempio `ExportDefinition`, `StrategyOnFull` e `Persistence`) per definire altre proprietà del flusso. Ciò include:

- L'obiettivoAWS IoT Analytics, Kinesis Data AWS IoT SiteWise Streams e le destinazioni Amazon S3 per le esportazioni automatiche. Per ulteriori informazioni, consulta the section called “Esporta le configurazioni per le destinazioni cloud supportate”.
- Priorità di esportazione. Stream manager esporta i flussi con priorità più alta prima dei flussi con priorità più bassa.
- Dimensione massima del batch e intervallo di batch per AWS IoT Analytics Kinesis Data Streams e destinazioni. AWS IoT SiteWise Stream manager esporta i messaggi quando viene soddisfatta una delle due condizioni.
- T ime-to-live (TTL). Il tempo necessario per garantire che i dati del flusso siano disponibili per l'elaborazione. È necessario assicurarsi che i dati possano essere utilizzati entro questo periodo di tempo. Questa non è una policy di eliminazione. È possibile che i dati non vengano eliminati immediatamente dopo il periodo TTL.
- Persistenza del flusso. Scegliere di salvare i flussi nel file system per mantenere i dati tra riavvii core o salvare i flussi in memoria.
- Numero di sequenza iniziale. Specificate il numero di sequenza del messaggio da utilizzare come messaggio iniziale nell'esportazione.

Per ulteriori informazioni su`MessageStreamDefinition`, consultate il riferimento SDK per la lingua di destinazione:

- MessageStreamDefinitionnell'SDK Java
- MessageStreamDefinitionnell'SDK Node.js
- MessageStreamDefinitionnell'SDK Python

> **Note**
>
> `StreamManagerClient`fornisce anche una destinazione di destinazione che è possibile utilizzare per esportare flussi su un server HTTP. Questo target è destinato esclusivamente a scopi di test. Non è stabile né è supportato per l'uso in ambienti di produzione.

Dopo aver creato uno stream, i componenti Greengrass possono aggiungere messaggi allo stream per inviare dati per l'esportazione e leggere i messaggi dallo stream per l'elaborazione locale. Il numero di flussi creati dipende dalle funzionalità hardware e dal business case. Una strategia consiste nel creare un flusso per ogni canale di destinazione nel AWS IoT Analytics nostro flusso di dati Kinesis, sebbene sia possibile definire più destinazioni per un flusso. Un flusso ha una lunga durata.

Requisiti

Questa operazione ha i seguenti requisiti:

- Versione minima dell'SDK di Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Esempi

Il frammento di codice seguente crea un flusso denominato `StreamName`. Definisce le proprietà dello `MessageStreamDefinition` stream nei tipi di dati subordinati.

Python

```
client = StreamManagerClient()

try:
    client.create_message_stream(MessageStreamDefinition(
        name="StreamName",    # Required.
        max_size=268435456,    # Default is 256 MB.
        stream_segment_size=16777216,    # Default is 16 MB.
        time_to_live_millis=None,    # By default, no TTL is enabled.
        strategy_on_full=StrategyOnFull.OverwriteOldestData,    # Required.
        persistence=Persistence.File,    # Default is File.
        flush_on_write=False,    # Default is false.
        export_definition=ExportDefinition(    # Optional. Choose where/how the
 stream is exported to the Cloud AWS.
```

```
            kinesis=None,
            iot_analytics=None,
            iot_sitewise=None,
            s3_task_executor=None
        )
    ))
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Riferimento all'SDK Python: create_message_stream | MessageStreamDefinition

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    client.createMessageStream(
            new MessageStreamDefinition()
                    .withName("StreamName") // Required.
                    .withMaxSize(268435456L)    // Default is 256 MB.
                    .withStreamSegmentSize(16777216L)    // Default is 16 MB.
                    .withTimeToLiveMillis(null)    // By default, no TTL is enabled.
                    .withStrategyOnFull(StrategyOnFull.OverwriteOldestData)    //
 Required.
                    .withPersistence(Persistence.File)    // Default is File.
                    .withFlushOnWrite(false)    // Default is false.
                    .withExportDefinition(    // Optional. Choose where/how the
 stream is exported to the Cloud AWS.
                            new ExportDefinition()
                                    .withKinesis(null)
                                    .withIotAnalytics(null)
                                    .withIotSitewise(null)
                                    .withS3(null)
                    )

    );
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Riferimento a Java createMessageStreamSDK: | MessageStreamDefinition

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
  try {
    await client.createMessageStream(
      new MessageStreamDefinition()
        .withName("StreamName") // Required.
        .withMaxSize(268435456)  // Default is 256 MB.
        .withStreamSegmentSize(16777216)  // Default is 16 MB.
        .withTimeToLiveMillis(null)  // By default, no TTL is enabled.
        .withStrategyOnFull(StrategyOnFull.OverwriteOldestData)  // Required.
        .withPersistence(Persistence.File)  // Default is File.
        .withFlushOnWrite(false)  // Default is false.
        .withExportDefinition(  // Optional. Choose where/how the stream is exported
 to the Cloud AWS.
          new ExportDefinition()
            .withKinesis(null)
            .withIotAnalytics(null)
            .withIotSiteWise(null)
            .withS3(null)
        )
    );
  } catch (e) {
    // Properly handle errors.
  }
});
client.onError((err) => {
  // Properly handle connection errors.
  // This is called only when the connection to the StreamManager server fails.
});
```

Riferimento all'SDK Node.js: | createMessageStreamMessageStreamDefinition

Per ulteriori informazioni sulla configurazione delle destinazioni di esportazione, consulta. the section called “Esporta le configurazioni per le destinazioni cloud supportate”

Aggiunta di un messaggio

Per inviare dati allo stream manager per l'esportazione, i componenti Greengrass aggiungono i dati allo stream di destinazione. La destinazione di esportazione determina il tipo di dati da passare a questo metodo.

Requisiti

Questa operazione presenta i seguenti requisiti:

- Versione minima dell'SDK di Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Esempi

AWS IoT Analyticso destinazioni di esportazione Kinesis Data Streams

Il frammento di codice seguente aggiunge un messaggio al flusso denominato `StreamName`. Per le AWS IoT Analytics nostre destinazioni Kinesis Data Streams, i componenti Greengrass aggiungono un blob di dati.

Questo frammento ha i seguenti requisiti:

- Versione minima dell'SDK di Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Python

```
client = StreamManagerClient()

try:
    sequence_number = client.append_message(stream_name="StreamName",
 data=b'Arbitrary bytes data')
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Riferimento Python SDK: append_message

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    long sequenceNumber = client.appendMessage("StreamName", "Arbitrary byte
 array".getBytes());
} catch (StreamManagerException e) {
```

```
    // Properly handle exception.
}
```

Riferimento Java SDK: AppendMessage

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const sequenceNumber = await client.appendMessage("StreamName",
 Buffer.from("Arbitrary byte array"));
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Riferimento SDK Node.js: appendMessage

AWS IoT SiteWisedestinazioni di esportazione

Il frammento di codice seguente aggiunge un messaggio al flusso denominato `StreamName`. Per le AWS IoT SiteWise destinazioni, i componenti Greengrass aggiungono un oggetto serializzato. `PutAssetPropertyValueEntry` Per ulteriori informazioni, consulta the section called "Esportazione in AWS IoT SiteWise".

> **Note**
>
> Quando si inviano dati aAWS IoT SiteWise, i dati devono soddisfare i requisiti dell'azione. `BatchPutAssetPropertyValue` Per ulteriori informazioni, consulta BatchPutAssetPropertyValue nella documentazione di riferimento dell'API AWS IoT SiteWise.

Questo frammento presenta i seguenti requisiti:

- Versione minima dell'SDK di Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Python

```
client = StreamManagerClient()

try:
    # SiteWise requires unique timestamps in all messages and also needs timestamps
 not earlier
    # than 10 minutes in the past. Add some randomness to time and offset.

    # Note: To create a new asset property data, you should use the classes defined
 in the
    # greengrasssdk.stream_manager module.

    time_in_nanos = TimeInNanos(
        time_in_seconds=calendar.timegm(time.gmtime()) - random.randint(0, 60),
 offset_in_nanos=random.randint(0, 10000)
    )
    variant = Variant(double_value=random.random())
    asset = [AssetPropertyValue(value=variant, quality=Quality.GOOD,
 timestamp=time_in_nanos)]
    putAssetPropertyValueEntry =
 PutAssetPropertyValueEntry(entry_id=str(uuid.uuid4()),
 property_alias="PropertyAlias", property_values=asset)
    sequence_number = client.append_message(stream_name="StreamName",
 Util.validate_and_serialize_to_json_bytes(putAssetPropertyValueEntry))
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Riferimento all'SDK Python: append_message | PutAssetPropertyValueEntry

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    Random rand = new Random();
    // Note: To create a new asset property data, you should use the classes defined
 in the
    // com.amazonaws.greengrass.streammanager.model.sitewise package.
    List<AssetPropertyValue> entries = new ArrayList<>() ;
```

```
    // IoTSiteWise requires unique timestamps in all messages and also needs
 timestamps not earlier
    // than 10 minutes in the past. Add some randomness to time and offset.
    final int maxTimeRandomness = 60;
    final int maxOffsetRandomness = 10000;
    double randomValue = rand.nextDouble();
    TimeInNanos timestamp = new TimeInNanos()
            .withTimeInSeconds(Instant.now().getEpochSecond() -
 rand.nextInt(maxTimeRandomness))
            .withOffsetInNanos((long) (rand.nextInt(maxOffsetRandomness)));
    AssetPropertyValue entry = new AssetPropertyValue()
            .withValue(new Variant().withDoubleValue(randomValue))
            .withQuality(Quality.GOOD)
            .withTimestamp(timestamp);
    entries.add(entry);

    PutAssetPropertyValueEntry putAssetPropertyValueEntry = new
 PutAssetPropertyValueEntry()
            .withEntryId(UUID.randomUUID().toString())
            .withPropertyAlias("PropertyAlias")
            .withPropertyValues(entries);
    long sequenceNumber = client.appendMessage("StreamName",
 ValidateAndSerialize.validateAndSerializeToJsonBytes(putAssetPropertyValueEntry));
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Riferimento Java SDK: [appendMessage | PutAssetPropertyValueEntry]

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const maxTimeRandomness = 60;
        const maxOffsetRandomness = 10000;
        const randomValue = Math.random();
        // Note: To create a new asset property data, you should use the classes
 defined in the
        // aws-greengrass-core-sdk StreamManager module.
        const timestamp = new TimeInNanos()
            .withTimeInSeconds(Math.round(Date.now() / 1000) -
 Math.floor(Math.random() - maxTimeRandomness))
            .withOffsetInNanos(Math.floor(Math.random() * maxOffsetRandomness));
```

```
        const entry = new AssetPropertyValue()
            .withValue(new Variant().withDoubleValue(randomValue))
            .withQuality(Quality.GOOD)
            .withTimestamp(timestamp);

        const putAssetPropertyValueEntry =    new PutAssetPropertyValueEntry()
            .withEntryId(`${ENTRY_ID_PREFIX}${i}`)
            .withPropertyAlias("PropertyAlias")
            .withPropertyValues([entry]);
        const sequenceNumber = await client.appendMessage("StreamName",
 util.validateAndSerializeToJsonBytes(putAssetPropertyValueEntry));
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Riferimento all'SDK Node.js: appendMessage | PutAssetPropertyValueEntry

Destinazioni di esportazione Amazon S3

Il seguente frammento aggiunge un'attività di esportazione allo stream denominato. `StreamName` Per le destinazioni Amazon S3, i componenti Greengrass aggiungono un oggetto serializzato che contiene informazioni sul file di input di origine e sull'`S3ExportTaskDefinition`oggetto Amazon S3 di destinazione. Se l'oggetto specificato non esiste, Stream Manager lo crea per te. Per ulteriori informazioni, consulta the section called “Esportazione su Amazon S3”.

Questo frammento ha i seguenti requisiti:

- Versione minima dell'SDK di Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Python

```
client = StreamManagerClient()

try:
    # Append an Amazon S3 Task definition and print the sequence number.
```

```
    s3_export_task_definition = S3ExportTaskDefinition(input_url="URLToFile",
 bucket="BucketName", key="KeyName")
    sequence_number = client.append_message(stream_name="StreamName",
 Util.validate_and_serialize_to_json_bytes(s3_export_task_definition))
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Riferimento all'SDK Python: append_message | S3 ExportTaskDefinition

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    // Append an Amazon S3 export task definition and print the sequence number.
    S3ExportTaskDefinition s3ExportTaskDefinition = new S3ExportTaskDefinition()
        .withBucket("BucketName")
        .withKey("KeyName")
        .withInputUrl("URLToFile");
    long sequenceNumber = client.appendMessage("StreamName",
 ValidateAndSerialize.validateAndSerializeToJsonBytes(s3ExportTaskDefinition));
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Riferimento a Java SDK: appendMessage | S3 ExportTaskDefinition

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
     // Append an Amazon S3 export task definition and print the sequence number.
     const taskDefinition = new S3ExportTaskDefinition()
        .withBucket("BucketName")
        .withKey("KeyName")
        .withInputUrl("URLToFile");
        const sequenceNumber = await client.appendMessage("StreamName",
 util.validateAndSerializeToJsonBytes(taskDefinition)));
    } catch (e) {
        // Properly handle errors.
```

```
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Riferimento all'SDK Node.js: appendMessage | S3 ExportTaskDefinition

Lettura di messaggi

Leggi i messaggi da uno stream.

Requisiti

Questa operazione ha i seguenti requisiti:

- Versione minima dell'SDK di Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Esempi

Il frammento di codice seguente legge i messaggi dal flusso denominato `StreamName`. Il metodo di lettura accetta un oggetto `ReadMessagesOptions` facoltativo che specifica il numero di sequenza da cui iniziare la lettura, i numeri minimo e massimo da leggere e un timeout per la lettura dei messaggi.

Python

```
client = StreamManagerClient()

try:
    message_list = client.read_messages(
        stream_name="StreamName",
        # By default, if no options are specified, it tries to read one message from
 the beginning of the stream.
        options=ReadMessagesOptions(
            desired_start_sequence_number=100,
            # Try to read from sequence number 100 or greater. By default, this is
 0.
            min_message_count=10,
```

```
            # Try to read 10 messages. If 10 messages are not available, then
 NotEnoughMessagesException is raised. By default, this is 1.
            max_message_count=100,    # Accept up to 100 messages. By default this
 is 1.
            read_timeout_millis=5000
            # Try to wait at most 5 seconds for the min_messsage_count to be
 fulfilled. By default, this is 0, which immediately returns the messages or an
 exception.
        )
    )
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

[Riferimento all'SDK Python: read_messages | ReadMessagesOptions]

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    List<Message> messages = client.readMessages("StreamName",
            // By default, if no options are specified, it tries to read one message
 from the beginning of the stream.
            new ReadMessagesOptions()
                    // Try to read from sequence number 100 or greater. By default
 this is 0.
                    .withDesiredStartSequenceNumber(100L)
                    // Try to read 10 messages. If 10 messages are not available,
 then NotEnoughMessagesException is raised. By default, this is 1.
                    .withMinMessageCount(10L)
                    // Accept up to 100 messages. By default this is 1.
                    .withMaxMessageCount(100L)
                    // Try to wait at most 5 seconds for the min_messsage_count to
 be fulfilled. By default, this is 0, which immediately returns the messages or an
 exception.
                    .withReadTimeoutMillis(Duration.ofSeconds(5L).toMillis())
    );
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Riferimento a Java SDK: ReadMessages | ReadMessagesOptions

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const messages = await client.readMessages("StreamName",
            // By default, if no options are specified, it tries to read one message
 from the beginning of the stream.
            new ReadMessagesOptions()
                // Try to read from sequence number 100 or greater. By default this
 is 0.
                .withDesiredStartSequenceNumber(100)
                // Try to read 10 messages. If 10 messages are not available, then
 NotEnoughMessagesException is thrown. By default, this is 1.
                .withMinMessageCount(10)
                // Accept up to 100 messages. By default this is 1.
                .withMaxMessageCount(100)
                // Try to wait at most 5 seconds for the minMessageCount to be
 fulfilled. By default, this is 0, which immediately returns the messages or an
 exception.
                .withReadTimeoutMillis(5 * 1000)
        );
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Riferimento all'SDK Node.js: ReadMessages | ReadMessagesOptions

Visualizzazione dell'elenco di flussi

Ottieni l'elenco degli stream nello stream manager.

Requisiti

Questa operazione ha i seguenti requisiti:

- Versione minima dell'SDK di Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Esempi

Il frammento di codice seguente ottiene un elenco dei flussi (per nome) in stream manager.

Python

```
client = StreamManagerClient()

try:
    stream_names = client.list_streams()
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Riferimento all'SDK Python: list_streams

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    List<String> streamNames = client.listStreams();
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Riferimento Java SDK: ListStreams

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const streams = await client.listStreams();
    } catch (e) {
        // Properly handle errors.
    }
});
```

```
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Riferimento all'SDK Node.js: ListStreams

Descrizione del flusso di messaggi

Ottieni i metadati relativi a uno stream, tra cui la definizione, la dimensione e lo stato dell'esportazione.

Requisiti

Questa operazione ha i seguenti requisiti:

- Versione minima dell'SDK di Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Esempi

Il frammento di codice seguente ottiene i metadati relativi al flusso denominato `StreamName`, inclusi la definizione, le dimensioni e gli stati di esportatore del flusso.

Python

```
client = StreamManagerClient()

try:
    stream_description = client.describe_message_stream(stream_name="StreamName")
    if stream_description.export_statuses[0].error_message:
        # The last export of export destination 0 failed with some error
        # Here is the last sequence number that was successfully exported
        stream_description.export_statuses[0].last_exported_sequence_number

    if (stream_description.storage_status.newest_sequence_number >
            stream_description.export_statuses[0].last_exported_sequence_number):
        pass
        # The end of the stream is ahead of the last exported sequence number
except StreamManagerException:
    pass
    # Properly handle errors.
```

```
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Riferimento all'SDK Python: describe_message_stream

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    MessageStreamInfo description = client.describeMessageStream("StreamName");
    String lastErrorMessage =
 description.getExportStatuses().get(0).getErrorMessage();
    if (lastErrorMessage != null && !lastErrorMessage.equals("")) {
        // The last export of export destination 0 failed with some error.
        // Here is the last sequence number that was successfully exported.
        description.getExportStatuses().get(0).getLastExportedSequenceNumber();
    }

    if (description.getStorageStatus().getNewestSequenceNumber() >
            description.getExportStatuses().get(0).getLastExportedSequenceNumber())
 {
        // The end of the stream is ahead of the last exported sequence number.
    }
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Riferimento a Java SDK: describeMessageStream

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const description = await client.describeMessageStream("StreamName");
        const lastErrorMessage = description.exportStatuses[0].errorMessage;
        if (lastErrorMessage) {
            // The last export of export destination 0 failed with some error.
            // Here is the last sequence number that was successfully exported.
            description.exportStatuses[0].lastExportedSequenceNumber;
        }

        if (description.storageStatus.newestSequenceNumber >
```

```
            description.exportStatuses[0].lastExportedSequenceNumber) {
            // The end of the stream is ahead of the last exported sequence number.
        }
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Riferimento all'SDK Node.js: describeMessageStream

Aggiorna il flusso di messaggi

Aggiorna le proprietà di uno stream esistente. Potresti voler aggiornare uno stream se i tuoi requisiti cambiano dopo la creazione dello stream. Per esempio:

- Aggiungi una nuova configurazione di esportazione per una Cloud AWS destinazione.
- Aumenta la dimensione massima di uno stream per modificare il modo in cui i dati vengono esportati o conservati. Ad esempio, la dimensione dello stream in combinazione con la tua strategia sulle impostazioni complete potrebbe comportare l'eliminazione o il rifiuto dei dati prima che lo stream manager possa elaborarli.
- Metti in pausa e riprendi le esportazioni, ad esempio se le attività di esportazione richiedono molto tempo e desideri razionare i dati di caricamento.

I componenti Greengrass seguono questo processo di alto livello per aggiornare uno stream:

1. Ottieni la descrizione dello stream.
2. Aggiorna le proprietà di destinazione sugli oggetti corrispondenti `MessageStreamDefinition` e subordinati.
3. Passa l'aggiornamento`MessageStreamDefinition`. Assicurati di includere le definizioni complete degli oggetti per lo stream aggiornato. Le proprietà non definite tornano ai valori predefiniti.

   È possibile specificare il numero di sequenza del messaggio da utilizzare come messaggio iniziale nell'esportazione.

Requisiti

Questa operazione presenta i seguenti requisiti:

- Versione minima dell'SDK di Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Esempi

Il seguente frammento aggiorna lo stream denominato. `StreamName` Aggiorna più proprietà di un flusso che esporta in Kinesis Data Streams.

Python

```
client = StreamManagerClient()

try:
    message_stream_info = client.describe_message_stream(STREAM_NAME)
    message_stream_info.definition.max_size=536870912
    message_stream_info.definition.stream_segment_size=33554432
    message_stream_info.definition.time_to_live_millis=3600000
    message_stream_info.definition.strategy_on_full=StrategyOnFull.RejectNewData
    message_stream_info.definition.persistence=Persistence.Memory
    message_stream_info.definition.flush_on_write=False
    message_stream_info.definition.export_definition.kinesis=
        [KinesisConfig(
            # Updating Export definition to add a Kinesis Stream configuration.
            identifier=str(uuid.uuid4()), kinesis_stream_name=str(uuid.uuid4()))]
    client.update_message_stream(message_stream_info.definition)
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Riferimento a Python SDK: | updateMessageStreamMessageStreamDefinition

Java

```
try (final StreamManagerClient client =
 GreengrassClientBuilder.streamManagerClient().build()) {
    MessageStreamInfo messageStreamInfo = client.describeMessageStream(STREAM_NAME);
    // Update the message stream with new values.
```

```
    client.updateMessageStream(
        messageStreamInfo.getDefinition()
            .withStrategyOnFull(StrategyOnFull.RejectNewData) // Required. Updating
 Strategy on full to reject new data.
           // Max Size update should be greater than initial Max Size defined in
 Create Message Stream request
            .withMaxSize(536870912L) // Update Max Size to 512 MB.
            .withStreamSegmentSize(33554432L) // Update Segment Size to 32 MB.
            .withFlushOnWrite(true) // Update flush on write to true.
            .withPersistence(Persistence.Memory) // Update the persistence to
 Memory.
            .withTimeToLiveMillis(3600000L)    // Update TTL to 1 hour.
            .withExportDefinition(
                // Optional. Choose where/how the stream is exported to the Cloud
 AWS.
                messageStreamInfo.getDefinition().getExportDefinition().
                    // Updating Export definition to add a Kinesis Stream
 configuration.
                    .withKinesis(new ArrayList<KinesisConfig>() {{
                        add(new KinesisConfig()
                            .withIdentifier(EXPORT_IDENTIFIER)
                            .withKinesisStreamName("test"));
                        }})
            );
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Riferimento a Java SDK: update_message_stream | MessageStreamDefinition

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        const messageStreamInfo = await c.describeMessageStream(STREAM_NAME);
        await client.updateMessageStream(
            messageStreamInfo.definition
                // Max Size update should be greater than initial Max Size defined
 in Create Message Stream request
                .withMaxSize(536870912)    // Default is 256 MB. Updating Max Size
 to 512 MB.
                .withStreamSegmentSize(33554432)    // Default is 16 MB. Updating
 Segment Size to 32 MB.
```

```
                .withTimeToLiveMillis(3600000)    // By default, no TTL is enabled.
 Update TTL to 1 hour.
                .withStrategyOnFull(StrategyOnFull.RejectNewData)    // Required.
 Updating Strategy on full to reject new data.
                .withPersistence(Persistence.Memory)    // Default is File. Update
 the persistence to Memory
                .withFlushOnWrite(true)    // Default is false. Updating to true.
                .withExportDefinition(
                    // Optional. Choose where/how the stream is exported to the
 Cloud AWS.
                    messageStreamInfo.definition.exportDefinition
                        // Updating Export definition to add a Kinesis Stream
 configuration.
                        .withKinesis([new
 KinesisConfig().withIdentifier(uuidv4()).withKinesisStreamName(uuidv4())])
                )
        );
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Riferimento all'updateMessageStreamSDK Node.js: | MessageStreamDefinition

Vincoli per l'aggiornamento degli stream

I seguenti vincoli si applicano all'aggiornamento dei flussi. A meno che non sia indicato nell'elenco seguente, gli aggiornamenti hanno effetto immediato.

- Non puoi aggiornare la persistenza di uno stream. Per modificare questo comportamento, elimina lo stream e crea uno stream che definisca la nuova politica di persistenza.
- Puoi aggiornare la dimensione massima di uno stream solo nelle seguenti condizioni:
  - La dimensione massima deve essere maggiore o uguale alla dimensione corrente dello stream. Per trovare queste informazioni, descrivi lo stream e poi controlla lo stato di archiviazione dell'`MessageStreamInfo`oggetto restituito.

  - La dimensione massima deve essere maggiore o uguale alla dimensione del segmento dello stream.
- Puoi aggiornare la dimensione del segmento di stream a un valore inferiore alla dimensione massima dello stream. L'impostazione aggiornata si applica ai nuovi segmenti.
- Gli aggiornamenti alla proprietà time to live (TTL) si applicano alle nuove operazioni di aggiunta. Se riduci questo valore, stream manager potrebbe anche eliminare i segmenti esistenti che superano il TTL.
- Gli aggiornamenti alla strategia sulla proprietà completa si applicano alle nuove operazioni di aggiunta. Se imposti la strategia per sovrascrivere i dati più vecchi, stream manager potrebbe anche sovrascrivere i segmenti esistenti in base alla nuova impostazione.
- Gli aggiornamenti alla proprietà flush on write si applicano ai nuovi messaggi.
- Gli aggiornamenti alle configurazioni di esportazione si applicano alle nuove esportazioni. La richiesta di aggiornamento deve includere tutte le configurazioni di esportazione che si desidera supportare. Altrimenti, stream manager le elimina.
  - Quando aggiorni una configurazione di esportazione, specifica l'identificatore della configurazione di esportazione di destinazione.
  - Per aggiungere una configurazione di esportazione, specificate un identificatore univoco per la nuova configurazione di esportazione.
  - Per eliminare una configurazione di esportazione, omettete la configurazione di esportazione.
- Per aggiornare il numero di sequenza iniziale di una configurazione di esportazione in uno stream, è necessario specificare un valore inferiore al numero di sequenza più recente. Per trovare queste informazioni, descrivi lo stream e quindi controlla lo stato di archiviazione dell'`MessageStreamInfo`oggetto restituito.

Eliminazione del flusso di messaggi

Elimina un flusso. Quando si elimina un flusso, tutti i dati memorizzati per il flusso vengono eliminati dal disco.

Requisiti

Questa operazione ha i seguenti requisiti:

- Versione minima dell'SDK di Stream Manager: Python: 1.1.0 | Java: 1.1.0 | Node.js: 1.1.0

Esempi

Il frammento di codice seguente elimina il flusso denominato `StreamName`.

Python

```
client = StreamManagerClient()

try:
    client.delete_message_stream(stream_name="StreamName")
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Riferimento all'SDK Python: deleteMessageStream

Java

```
try (final StreamManagerClient client =
 StreamManagerClientFactory.standard().build()) {
    client.deleteMessageStream("StreamName");
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Riferimento Java SDK: delete_message_stream

Node.js

```
const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        await client.deleteMessageStream("StreamName");
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
```

```
});
```

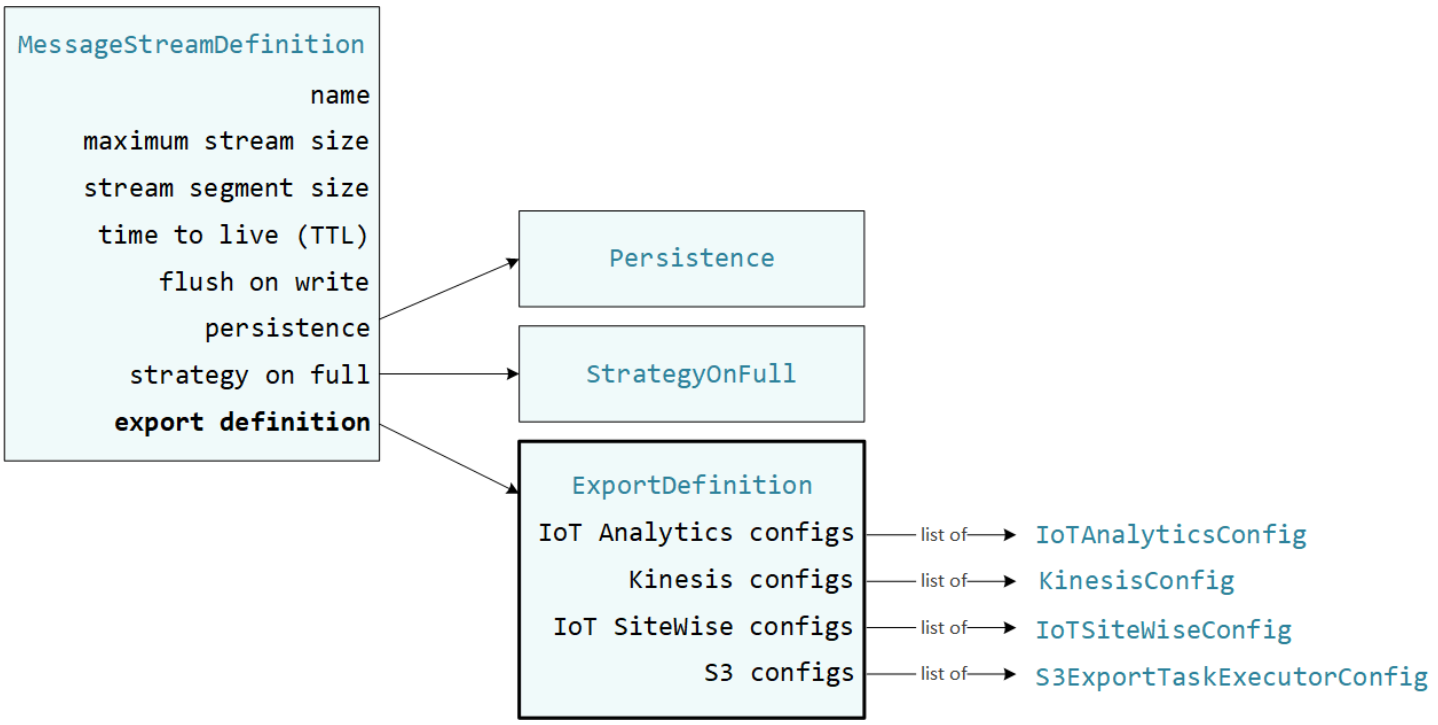
Riferimento all'SDK Node.js: deleteMessageStream

Consulta anche

- Gestisci i flussi di dati sui dispositivi core Greengrass
- Configurazione di Stream Manager di AWS IoT Greengrass
- Esporta configurazioni per le destinazioni supportate Cloud AWS
- `StreamManagerClient`nel riferimento all'SDK di Stream Manager:
  - Python
  - Java
  - Node.js

Esporta configurazioni per le destinazioni supportate Cloud AWS

I componenti Greengrass definiti dall'utente vengono `StreamManagerClient` utilizzati nell'SDK Stream Manager per interagire con lo stream manager. Quando un componente crea uno stream o aggiorna uno stream, passa un `MessageStreamDefinition` oggetto che rappresenta le proprietà dello stream, inclusa la definizione di esportazione. L'`ExportDefinition`oggetto contiene le configurazioni di esportazione definite per lo stream. Stream Manager utilizza queste configurazioni di esportazione per determinare dove e come esportare lo stream.



È possibile definire zero o più configurazioni di esportazione su uno stream, incluse più configurazioni di esportazione per un singolo tipo di destinazione. Ad esempio, puoi esportare uno stream su due AWS IoT Analytics canali e un flusso di dati Kinesis.

In caso di tentativi di esportazione falliti, stream manager riprova continuamente a esportare i dati verso il a Cloud AWS intervalli fino a cinque minuti. Il numero di tentativi di nuovo tentativo non ha un limite massimo.

> **Note**
>
> `StreamManagerClient`fornisce anche una destinazione di destinazione che è possibile utilizzare per esportare i flussi su un server HTTP. Questo target è destinato esclusivamente a scopi di test. Non è stabile né è supportato per l'uso in ambienti di produzione.

Cloud AWSDestinazioni supportate

- Canali AWS IoT Analytics
- Flussi di dati Amazon Kinesis
- AWS IoT SiteWiseproprietà degli asset
- Oggetti Amazon S3

Sei responsabile del mantenimento di queste Cloud AWS risorse.

Canali AWS IoT Analytics

Stream manager supporta le esportazioni automatiche versoAWS IoT Analytics. AWS IoT Analyticsconsente di eseguire analisi avanzate sui dati per aiutare a prendere decisioni aziendali e migliorare i modelli di apprendimento automatico. Per ulteriori informazioni, consulta Cos'èAWS IoT Analytics? nella Guida AWS IoT Analytics per l'utente.

Nell'SDK Stream Manager, i componenti Greengrass utilizzano per definire `IoTAnalyticsConfig` la configurazione di esportazione per questo tipo di destinazione. Per ulteriori informazioni, consultate il riferimento SDK per la lingua di destinazione:

- IoT AnalyticsConfig nell'SDK Python
- IoT AnalyticsConfig nell'SDK Java
- IoT AnalyticsConfig nell'SDK Node.js

Requisiti

Questa destinazione di esportazione presenta i seguenti requisiti:

- I canali di destinazione in ingresso AWS IoT Analytics devono trovarsi nello stesso Account AWS dispositivo principale Greengrass. Regione AWS
- Autorizza i dispositivi principali a interagire conAWSserviziDevono consentire l'`iotanalytics:BatchPutMessage`autorizzazione per i canali di destinazione. Per esempio:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iotanalytics:BatchPutMessage"
      ],
      "Resource": [
        "arn:aws:iotanalytics:region:account-id:channel/channel_1_name",
        "arn:aws:iotanalytics:region:account-id:channel/channel_2_name"
      ]
    }
  ]
```

```
}
```

È possibile concedere un accesso granulare o condizionale alle risorse, ad esempio utilizzando uno schema di denominazione con caratteri jolly*. Per ulteriori informazioni, consulta Aggiungere e rimuovere le policy IAM nella IAM User Guide.

Esportazione in AWS IoT Analytics

Per creare un flusso che esporta inAWS IoT Analytics, i componenti Greengrass creano un flusso con una definizione di esportazione che include uno o più `IoTAnalyticsConfig` oggetti. Questo oggetto definisce le impostazioni di esportazione, come il canale di destinazione, la dimensione del batch, l'intervallo del batch e la priorità.

Quando i componenti Greengrass ricevono dati dai dispositivi, aggiungono messaggi che contengono un blob di dati allo stream di destinazione.

Quindi, stream manager esporta i dati in base alle impostazioni del batch e alla priorità definite nelle configurazioni di esportazione dello stream.

Flussi di dati Amazon Kinesis

Stream Manager supporta le esportazioni automatiche verso Amazon Kinesis Data Streams. Kinesis Data Streams viene comunemente utilizzato per aggregare dati di grandi volumi e caricarli in un data warehouse o cluster. MapReduce Per ulteriori informazioni, consulta Cos'è Amazon Kinesis Data Streams? nella Amazon Kinesis Developer Guide.

Nell'SDK Stream Manager, i componenti Greengrass utilizzano per definire `KinesisConfig` la configurazione di esportazione per questo tipo di destinazione. Per ulteriori informazioni, consultate il riferimento SDK per la lingua di destinazione:

- KinesisConfignell'SDK Python
- KinesisConfignell'SDK Java
- KinesisConfignell'SDK Node.js

Requisiti

Questa destinazione di esportazione presenta i seguenti requisiti:

- I flussi di destinazione in Kinesis Data Streams devono trovarsi nello Regione AWS stesso dispositivo principale Account AWS Greengrass.
- Autorizza i dispositivi principali a interagire conAWSserviziDevono consentire l'`kinesis:PutRecords`autorizzazione per indirizzare i flussi di dati. Per esempio:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "kinesis:PutRecords"
      ],
      "Resource": [
        "arn:aws:kinesis:region:account-id:stream/stream_1_name",
        "arn:aws:kinesis:region:account-id:stream/stream_2_name"
      ]
    }
  ]
}
```

  È possibile concedere un accesso granulare o condizionale alle risorse, ad esempio utilizzando uno schema di denominazione con caratteri jolly*. Per ulteriori informazioni, consulta Aggiungere e rimuovere le policy IAM nella IAM User Guide.

Esportazione in Kinesis Data Streams

Per creare un flusso che esporta in Kinesis Data Streams, i componenti Greengrass creano un flusso con una definizione di esportazione che include uno o più oggetti. `KinesisConfig` Questo oggetto definisce le impostazioni di esportazione, come il flusso di dati di destinazione, la dimensione del batch, l'intervallo del batch e la priorità.

Quando i componenti Greengrass ricevono dati dai dispositivi, aggiungono messaggi che contengono un blob di dati allo stream di destinazione. Quindi, stream manager esporta i dati in base alle impostazioni del batch e alla priorità definite nelle configurazioni di esportazione dello stream.

Stream Manager genera un UUID univoco e casuale come chiave di partizione per ogni record caricato su Amazon Kinesis.

AWS IoT SiteWiseproprietà degli asset

Stream manager supporta le esportazioni automatiche versoAWS IoT SiteWise. AWS IoT SiteWiseconsente di raccogliere, organizzare e analizzare dati provenienti da apparecchiature industriali su larga scala. Per ulteriori informazioni, consulta Cos'èAWS IoT SiteWise? nella Guida AWS IoT SiteWise per l'utente.

Nell'SDK Stream Manager, i componenti Greengrass utilizzano per definire `IoTSiteWiseConfig` la configurazione di esportazione per questo tipo di destinazione. Per ulteriori informazioni, consultate il riferimento SDK per la lingua di destinazione:

- IoT SiteWiseConfig nell'SDK Python
- IoT SiteWiseConfig nell'SDK Java
- IoT SiteWiseConfig nell'SDK Node.js

> **Note**
>
> AWSfornisce anche AWS IoT SiteWise componenti che offrono una soluzione predefinita che è possibile utilizzare per lo streaming di dati da fonti OPC-UA. Per ulteriori informazioni, consulta Collettore IoT SiteWise OPC-UA.

Requisiti

Questa destinazione di esportazione presenta i seguenti requisiti:

- Le proprietà degli asset di destinazione AWS IoT SiteWise devono trovarsi nello stesso Account AWS Regione AWS dispositivo principale di Greengrass.

  > **Note**
  >
  > Per l'elenco dei sistemi AWS IoT SiteWise supportati, consulta Regione AWS gli AWS IoT SiteWiseendpoint e le quote nella AWS Guida generale.

- Autorizza i dispositivi principali a interagire conAWSserviziDevono consentire l'`iotsitewise:BatchPutAssetPropertyValue`autorizzazione per indirizzare le proprietà degli asset. La seguente politica di esempio utilizza la chiave `iotsitewise:assetHierarchyPath` condition per concedere l'accesso a una risorsa principale di destinazione e ai suoi figli. Puoi

rimuoverla `Condition` dalla policy per consentire l'accesso a tutte le tue AWS IoT SiteWise risorse o specificare gli ARN di singole risorse.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
       "Effect": "Allow",
       "Action": "iotsitewise:BatchPutAssetPropertyValue",
       "Resource": "*",
       "Condition": {
         "StringLike": {
           "iotsitewise:assetHierarchyPath": [
             "/root node asset ID",
             "/root node asset ID/*"
           ]
         }
       }
    }
  ]
}
```

È possibile concedere un accesso granulare o condizionale alle risorse, ad esempio utilizzando uno schema di denominazione con caratteri jolly*. Per ulteriori informazioni, consulta Aggiungere e rimuovere le policy IAM nella IAM User Guide.

Per importanti informazioni sulla sicurezza, consulta BatchPutAssetPropertyValue l'autorizzazione nella Guida AWS IoT SiteWise per l'utente.

Esportazione in AWS IoT SiteWise

Per creare un flusso che esporta inAWS IoT SiteWise, i componenti Greengrass creano un flusso con una definizione di esportazione che include uno o più `IoTSiteWiseConfig` oggetti. Questo oggetto definisce le impostazioni di esportazione, come la dimensione del batch, l'intervallo del batch e la priorità.

Quando i componenti Greengrass ricevono dati sulle proprietà degli asset dai dispositivi, aggiungono messaggi che contengono i dati allo stream di destinazione. I messaggi sono `PutAssetPropertyValueEntry` oggetti serializzati in formato JSON che contengono valori di

proprietà per una o più proprietà degli asset. Per ulteriori informazioni, consulta Aggiungi messaggio per le destinazioni di esportazione. AWS IoT SiteWise

> **Note**
>
> Quando invii dati aAWS IoT SiteWise, i tuoi dati devono soddisfare i requisiti dell'`BatchPutAssetPropertyValue`azione. Per ulteriori informazioni, consulta BatchPutAssetPropertyValue nella documentazione di riferimento dell'API AWS IoT SiteWise.

Quindi, stream manager esporta i dati in base alle impostazioni del batch e alla priorità definite nelle configurazioni di esportazione dello stream.

Puoi modificare le impostazioni dello stream manager e la logica dei componenti Greengrass per progettare la tua strategia di esportazione. Per esempio:

- Per esportazioni quasi in tempo reale, impostate impostazioni ridotte per la dimensione del batch e l'intervallo e aggiungete i dati allo stream quando vengono ricevuti.
- Per ottimizzare il batching, mitigare i vincoli di larghezza di banda o ridurre al minimo i costi, i componenti Greengrass possono raggruppare i punti dati timestamp-quality-value (TQV) ricevuti per una singola proprietà dell'asset prima di aggiungere i dati allo stream. Una strategia consiste nell'inserire in un unico messaggio le voci relative a un massimo di 10 diverse combinazioni proprietà-asset, o alias di proprietà, anziché inviare più di una voce per la stessa proprietà. Questo aiuta lo stream manager a rimanere entro le quote. AWS IoT SiteWise

Oggetti Amazon S3

Stream Manager supporta le esportazioni automatiche verso Amazon S3. Puoi usare Amazon S3 per archiviare e recuperare grandi quantità di dati. Per ulteriori informazioni, consulta Cos'è Amazon S3? nella Guida per sviluppatori di Amazon Simple Storage Service.

Nell'SDK Stream Manager, i componenti Greengrass utilizzano per definire `S3ExportTaskExecutorConfig` la configurazione di esportazione per questo tipo di destinazione. Per ulteriori informazioni, consultate il riferimento SDK per la lingua di destinazione:

- S3 ExportTaskExecutorConfig nell'SDK Python
- S3 nell'SDK Java ExportTaskExecutorConfig
- S3 ExportTaskExecutorConfig nell'SDK Node.js

Requisiti

Questa destinazione di esportazione presenta i seguenti requisiti:

- I bucket Amazon S3 di Target devono trovarsi nello stesso dispositivo Account AWS principale Greengrass.
- Se una funzione Lambda eseguita in modalità contenitore Greengrass scrive file di input in una directory di file di input, è necessario montare la directory come volume nel contenitore con autorizzazioni di scrittura. Ciò garantisce che i file vengano scritti nel file system root e siano visibili al componente stream manager, che viene eseguito all'esterno del contenitore.
- Se un componente del contenitore Docker scrive file di input in una directory di file di input, è necessario montare la directory come volume nel contenitore con autorizzazioni di scrittura. Ciò garantisce che i file vengano scritti nel file system root e siano visibili al componente stream manager, che viene eseguito all'esterno del contenitore.
- Autorizza i dispositivi principali a interagire conAWSserviziDeve consentire le seguenti autorizzazioni per i bucket di destinazione. Per esempio:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:AbortMultipartUpload",
        "s3:ListMultipartUploadParts"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-1-name/*",
        "arn:aws:s3:::bucket-2-name/*"
      ]
    }
  ]
}
```

  È possibile concedere un accesso granulare o condizionale alle risorse, ad esempio utilizzando uno schema di denominazione con caratteri jolly. * Per ulteriori informazioni, consulta Aggiungere e rimuovere le policy IAM nella IAM User Guide.

Esportazione su Amazon S3

Per creare uno stream che esporta in Amazon S3, i componenti Greengrass utilizzano l'`S3ExportTaskExecutorConfig`oggetto per configurare la politica di esportazione. La policy definisce le impostazioni di esportazione, come la soglia e la priorità di caricamento multiparte. Per le esportazioni di Amazon S3, stream manager carica i dati che legge dai file locali sul dispositivo principale. Per avviare un caricamento, i componenti Greengrass aggiungono un'attività di esportazione allo stream di destinazione. L'attività di esportazione contiene informazioni sul file di input e sull'oggetto Amazon S3 di destinazione. Stream Manager esegue le attività nella sequenza in cui vengono aggiunte allo stream.

> **Note**
>
> Il bucket di destinazione deve già esistere nel tuo. Account AWS Se non esiste un oggetto per la chiave specificata, lo stream manager crea l'oggetto per te.

Stream manager utilizza la proprietà della soglia di caricamento multiparte, l'impostazione della dimensione minima della parte e la dimensione del file di input per determinare come caricare i dati. La soglia di caricamento in più parti deve essere maggiore o uguale alla dimensione minima della parte. Se desideri caricare dati in parallelo, puoi creare più stream.

Le chiavi che specificano gli oggetti Amazon S3 di destinazione possono includere DateTimeFormatter stringhe Java valide nei segnaposto. `!{timestamp:value}` Puoi utilizzare questi segnaposto con timestamp per partizionare i dati in Amazon S3 in base all'ora in cui i dati del file di input sono stati caricati. Ad esempio, il seguente nome di chiave si risolve in un valore come. `my-key/2020/12/31/data.txt`

```
my-key/!{timestamp:YYYY}/!{timestamp:MM}/!{timestamp:dd}/data.txt
```

> **Note**
>
> Se desideri monitorare lo stato dell'esportazione di uno stream, crea prima uno stream di stato e poi configura il flusso di esportazione per utilizzarlo. Per ulteriori informazioni, consulta the section called “Monitora le attività di esportazione”.

Gestisci i dati di input

Puoi creare codice utilizzato dalle applicazioni IoT per gestire il ciclo di vita dei dati di input. Il seguente flusso di lavoro di esempio mostra come utilizzare i componenti Greengrass per gestire questi dati.

1. Un processo locale riceve dati da dispositivi o periferiche e quindi li scrive su file in una directory sul dispositivo principale. Questi sono i file di input per lo stream manager.
2. Un componente Greengrass analizza la directory e aggiunge un'attività di esportazione allo stream di destinazione quando viene creato un nuovo file. L'attività è un `S3ExportTaskDefinition` oggetto serializzato in JSON che specifica l'URL del file di input, il bucket e la chiave Amazon S3 di destinazione e i metadati utente opzionali.
3. Stream Manager legge il file di input ed esporta i dati in Amazon S3 nell'ordine delle attività aggiunte. Il bucket di destinazione deve già esistere nel tuo. Account AWS Se non esiste un oggetto per la chiave specificata, lo stream manager crea l'oggetto per te.
4. Il componente Greengrass legge i messaggi da un flusso di stato per monitorare lo stato dell'esportazione. Una volta completate le attività di esportazione, il componente Greengrass può eliminare i file di input corrispondenti. Per ulteriori informazioni, consulta the section called “Monitora le attività di esportazione”.

Monitora le attività di esportazione

Puoi creare codice utilizzato dalle applicazioni IoT per monitorare lo stato delle esportazioni Amazon S3. I componenti Greengrass devono creare un flusso di stato e quindi configurare il flusso di esportazione per scrivere aggiornamenti di stato nel flusso di stato. Un singolo flusso di stato può ricevere aggiornamenti di stato da più flussi esportati in Amazon S3.

Innanzitutto, crea uno stream da utilizzare come flusso di stato. Puoi configurare le dimensioni e le politiche di conservazione dello stream per controllare la durata dei messaggi di stato. Per esempio:

- `Persistence`Imposta su `Memory` se non desideri archiviare i messaggi di stato.
- Impostato `StrategyOnFull` in `OverwriteOldestData` modo che i nuovi messaggi di stato non vadano persi.

Quindi, crea o aggiorna il flusso di esportazione per utilizzare il flusso di stato. In particolare, imposta la proprietà di configurazione dello stato della configurazione di `S3ExportTaskExecutorConfig` esportazione dello stream. Questa impostazione indica al gestore dello stream di scrivere messaggi

di stato sulle attività di esportazione nello stream di stato. Nell'`StatusConfig`oggetto, specificate il nome del flusso di stato e il livello di verbosità. I seguenti valori supportati vanno da least verbose (`ERROR`) a most verbose (). `TRACE` Il valore predefinito è `INFO`.

- `ERROR`
- `WARN`
- `INFO`
- `DEBUG`
- `TRACE`

Il seguente flusso di lavoro di esempio mostra come i componenti Greengrass potrebbero utilizzare un flusso di stato per monitorare lo stato dell'esportazione.

1. Come descritto nel flusso di lavoro precedente, un componente Greengrass aggiunge un'attività di esportazione a uno stream configurato per scrivere messaggi di stato sulle attività di esportazione in un flusso di stato. L'operazione di aggiunta restituisce un numero di sequenza che rappresenta l'ID dell'attività.
2. Un componente Greengrass legge i messaggi in sequenza dallo stream di stato, quindi filtra i messaggi in base al nome del flusso e all'ID dell'attività o in base a una proprietà dell'attività di esportazione dal contesto del messaggio. Ad esempio, il componente Greengrass può filtrare in base all'URL del file di input dell'attività di esportazione, che è rappresentato dall'`S3ExportTaskDefinition`oggetto nel contesto del messaggio.

   I seguenti codici di stato indicano che un'attività di esportazione ha raggiunto lo stato di completamento:

   - `Success`. Il caricamento è stato completato con successo.
   - `Failure`. Lo stream manager ha riscontrato un errore, ad esempio, il bucket specificato non esiste. Dopo aver risolto il problema, puoi aggiungere nuovamente l'attività di esportazione allo stream.
   - `Canceled`. L'attività è stata interrotta perché la definizione dello stream o dell'esportazione è stata eliminata o il periodo time-to-live (TTL) dell'attività è scaduto.

> **Note**
>
> L'attività potrebbe anche avere lo stato o. `InProgress` `Warning` Stream manager emette avvisi quando un evento restituisce un errore che non influisce sull'esecuzione dell'attività. Ad esempio, la mancata pulizia di un caricamento parziale restituisce un avviso.

3. Una volta completate le attività di esportazione, il componente Greengrass può eliminare i file di input corrispondenti.

L'esempio seguente mostra come un componente Greengrass potrebbe leggere ed elaborare i messaggi di stato.

Python

```
import time
from stream_manager import (
    ReadMessagesOptions,
    Status,
    StatusConfig,
    StatusLevel,
    StatusMessage,
    StreamManagerClient,
)
from stream_manager.util import Util

client = StreamManagerClient()

try:
    # Read the statuses from the export status stream
    is_file_uploaded_to_s3 = False
    while not is_file_uploaded_to_s3:
        try:
            messages_list = client.read_messages(
                "StatusStreamName", ReadMessagesOptions(min_message_count=1,
 read_timeout_millis=1000)
            )
            for message in messages_list:
                # Deserialize the status message first.
                status_message = Util.deserialize_json_bytes_to_obj(message.payload,
 StatusMessage)
```

```
                # Check the status of the status message. If the status is
 "Success",
                # the file was successfully uploaded to S3.
                # If the status was either "Failure" or "Cancelled", the server was
 unable to upload the file to S3.
                # We will print the message for why the upload to S3 failed from the
 status message.
                # If the status was "InProgress", the status indicates that the
 server has started uploading
                # the S3 task.
                if status_message.status == Status.Success:
                    logger.info("Successfully uploaded file at path " + file_url + "
 to S3.")
                    is_file_uploaded_to_s3 = True
                elif status_message.status == Status.Failure or
 status_message.status == Status.Canceled:
                    logger.info(
                        "Unable to upload file at path " + file_url + " to S3.
 Message: " + status_message.message
                    )
                    is_file_uploaded_to_s3 = True
            time.sleep(5)
        except StreamManagerException:
            logger.exception("Exception while running")
except StreamManagerException:
    pass
    # Properly handle errors.
except ConnectionError or asyncio.TimeoutError:
    pass
    # Properly handle errors.
```

Riferimento all'SDK Python: read_messages | StatusMessage

Java

```
import com.amazonaws.greengrass.streammanager.client.StreamManagerClient;
import com.amazonaws.greengrass.streammanager.client.StreamManagerClientFactory;
import com.amazonaws.greengrass.streammanager.client.utils.ValidateAndSerialize;
import com.amazonaws.greengrass.streammanager.model.ReadMessagesOptions;
import com.amazonaws.greengrass.streammanager.model.Status;
import com.amazonaws.greengrass.streammanager.model.StatusConfig;
import com.amazonaws.greengrass.streammanager.model.StatusLevel;
import com.amazonaws.greengrass.streammanager.model.StatusMessage;
```

```
try (final StreamManagerClient client =
StreamManagerClientFactory.standard().build()) {
   try {
       boolean isS3UploadComplete = false;
       while (!isS3UploadComplete) {
           try {
               // Read the statuses from the export status stream
               List<Message> messages = client.readMessages("StatusStreamName",
                   new
ReadMessagesOptions().withMinMessageCount(1L).withReadTimeoutMillis(1000L));
               for (Message message : messages) {
                   // Deserialize the status message first.
                   StatusMessage statusMessage =
ValidateAndSerialize.deserializeJsonBytesToObj(message.getPayload(),
StatusMessage.class);
                   // Check the status of the status message. If the status is
"Success", the file was successfully uploaded to S3.
                   // If the status was either "Failure" or "Canceled", the server
was unable to upload the file to S3.
                   // We will print the message for why the upload to S3 failed
from the status message.
                   // If the status was "InProgress", the status indicates that the
server has started uploading the S3 task.
                   if (Status.Success.equals(statusMessage.getStatus())) {
                       System.out.println("Successfully uploaded file at path " +
FILE_URL + " to S3.");
                       isS3UploadComplete = true;
                    } else if (Status.Failure.equals(statusMessage.getStatus()) ||
Status.Canceled.equals(statusMessage.getStatus())) {
                       System.out.println(String.format("Unable to upload file at
path %s to S3. Message %s",

statusMessage.getStatusContext().getS3ExportTaskDefinition().getInputUrl(),
                           statusMessage.getMessage()));
                       sS3UploadComplete = true;
                   }
               }
           } catch (StreamManagerException ignored) {
           } finally {
               // Sleep for sometime for the S3 upload task to complete before
trying to read the status message.
               Thread.sleep(5000);
           }
       } catch (e) {
```

```
        // Properly handle errors.
    }
} catch (StreamManagerException e) {
    // Properly handle exception.
}
```

Riferimento Java SDK: ReadMessages | StatusMessage

Node.js

```
const {
    StreamManagerClient, ReadMessagesOptions,
    Status, StatusConfig, StatusLevel, StatusMessage,
    util,
} = require(*'aws-greengrass-stream-manager-sdk'*);

const client = new StreamManagerClient();
client.onConnected(async () => {
    try {
        let isS3UploadComplete = false;
        while (!isS3UploadComplete) {
            try {
                // Read the statuses from the export status stream
                const messages = await c.readMessages("StatusStreamName",
                    new ReadMessagesOptions()
                        .withMinMessageCount(1)
                        .withReadTimeoutMillis(1000));

                messages.forEach((message) => {
                    // Deserialize the status message first.
                    const statusMessage =
 util.deserializeJsonBytesToObj(message.payload, StatusMessage);
                    // Check the status of the status message. If the status is
 'Success', the file was successfully uploaded to S3.
                    // If the status was either 'Failure' or 'Cancelled', the server
 was unable to upload the file to S3.
                    // We will print the message for why the upload to S3 failed
 from the status message.
                    // If the status was "InProgress", the status indicates that the
 server has started uploading the S3 task.
                    if (statusMessage.status === Status.Success) {
                        console.log(`Successfully uploaded file at path ${FILE_URL}
 to S3.`);
                        isS3UploadComplete = true;
```

```
                    } else if (statusMessage.status === Status.Failure ||
 statusMessage.status === Status.Canceled) {
                        console.log(`Unable to upload file at path ${FILE_URL} to
 S3. Message: ${statusMessage.message}`);
                        isS3UploadComplete = true;
                    }
                });
                // Sleep for sometime for the S3 upload task to complete before
 trying to read the status message.
                await new Promise((r) => setTimeout(r, 5000));
            } catch (e) {
                // Ignored
            }
    } catch (e) {
        // Properly handle errors.
    }
});
client.onError((err) => {
    // Properly handle connection errors.
    // This is called only when the connection to the StreamManager server fails.
});
```

Riferimento all'SDK Node.js: ReadMessages | StatusMessage

Configurazione di Stream Manager di AWS IoT Greengrass

Sui dispositivi core Greengrass, lo stream manager può archiviare, elaborare ed esportare i dati dei dispositivi IoT. Stream manager fornisce i parametri che potete utilizzare per configurare le impostazioni di runtime. Queste impostazioni si applicano a tutti gli stream sul dispositivo principale Greengrass. È possibile utilizzare la AWS IoT Greengrass console o l'API per configurare le impostazioni dello stream manager quando si distribuisce il componente. Le modifiche hanno effetto dopo il completamento della distribuzione.

Parametri di Stream Manager

Stream Manager fornisce i seguenti parametri che è possibile configurare quando si distribuisce il componente sui dispositivi principali. Tutti i parametri sono opzionali:

Directory di storage

Nome parametro: `STREAM_MANAGER_STORE_ROOT_DIR`

Il percorso assoluto della cartella locale utilizzata per archiviare gli stream. Questo valore deve iniziare con una barra (ad esempio, `/data`).

È necessario specificare una cartella esistente e l'utente di sistema che esegue il componente stream manager deve disporre delle autorizzazioni per leggere e scrivere in questa cartella. Ad esempio, è possibile eseguire i seguenti comandi per creare e configurare una cartella `/var/greengrass/streams`, specificata come cartella principale dello stream manager. Questi comandi consentono all'utente di sistema predefinito di leggere e scrivere in questa cartella. `ggc_user`

```
sudo mkdir /var/greengrass/streams
sudo chown ggc_user /var/greengrass/streams
sudo chmod 700 /var/greengrass/streams
```

Per informazioni relative alla protezione dei dati del flusso, consulta the section called “Sicurezza dei dati locali”.

Impostazione predefinita: */greengrass/v2*`/work/aws.greengrass.StreamManager`

Porta del server

Nome parametro: `STREAM_MANAGER_SERVER_PORT`

Il numero di porta locale utilizzato per comunicare con stream manager. Il valore predefinito è `8088`.

È possibile specificare `0` di utilizzare una porta disponibile in modo casuale.

Autentica client

Nome parametro: `STREAM_MANAGER_AUTHENTICATE_CLIENT`

Indica se i client devono essere autenticati per interagire con stream manager. Tutte le interazioni tra i client e lo stream manager sono controllate dallo Stream Manager SDK. Questo parametro determina quali client possono chiamare l'SDK Stream Manager per lavorare con gli stream. Per ulteriori informazioni, consulta the section called “Autenticazione client”.

I valori validi sono `true` e `false`. Il valore predefinito è `true` (consigliato).

- `true`. Consente solo i componenti Greengrass come client. I componenti utilizzano i protocolli AWS IoT Greengrass Core interni per l'autenticazione con Stream Manager SDK.

- `false`. Consente a qualsiasi processo eseguito sul AWS IoT Greengrass Core di essere un client. Non impostate il valore a `false` meno che il vostro business case non lo richieda. Ad esempio, utilizzare `false` solo se i processi non componenti sul dispositivo principale devono comunicare direttamente con lo stream manager.

Larghezza di banda massima

Nome parametro: `STREAM_MANAGER_EXPORTER_MAX_BANDWIDTH`

La larghezza di banda massima media (in kilobit al secondo) che può essere utilizzata per esportare i dati. L'impostazione predefinita consente l'uso illimitato della larghezza di banda disponibile.

Dimensione del pool di thread

Nome parametro: `STREAM_MANAGER_EXPORTER_THREAD_POOL_SIZE`

Il numero massimo di thread attivi che possono essere utilizzati per esportare i dati. Il valore predefinito è `5`.

La dimensione ottimale dipende dall'hardware, dal volume del flusso e dal numero pianificato di flussi di esportazione. Se la velocità di esportazione è bassa, puoi regolare questa impostazione per trovare la dimensione ottimale per l'hardware e il business case. La CPU e la memoria dell'hardware del dispositivo core sono fattori limitanti. Per iniziare, è possibile provare a impostare questo valore uguale al numero di core di processore sul dispositivo.

Fare attenzione a non impostare una dimensione superiore a quella supportata dall'hardware. Ogni stream consuma risorse hardware, quindi cercate di limitare il numero di flussi di esportazione su dispositivi con restrizioni.

Argomenti JVM

Nome parametro: `JVM_ARGS`

Argomenti Java Virtual Machine personalizzati da passare a Stream Manager all'avvio. Più argomenti devono essere separati da spazi.

Utilizza questo parametro solo quando devi sostituire le impostazioni predefinite utilizzate dalla JVM. Ad esempio, potrebbe essere necessario aumentare la dimensione heap predefinita se prevedi di esportare un numero elevato di flussi.

Livello di logging

Nome parametro: `LOG_LEVEL`

Il livello di registrazione per il componente. Scegliete tra i seguenti livelli di registro, elencati qui in ordine di livello:

- `TRACE`
- `DEBUG`
- `INFO`
- `WARN`
- `ERROR`

Impostazione predefinita: `INFO`

Dimensione minima per il caricamento in più parti

Nome parametro: `STREAM_MANAGER_EXPORTER_S3_DESTINATION_MULTIPART_UPLOAD_MIN_PART_SIZE_BYTES`

La dimensione minima (in byte) di una parte in un caricamento multiparte su Amazon S3. Stream Manager utilizza questa impostazione e la dimensione del file di input per determinare come raggruppare i dati in una richiesta PUT composta da più parti. Il valore minimo e predefinito è di `5242880` byte (5 MB).

> **Note**
>
> Stream Manager utilizza la `sizeThresholdForMultipartUploadBytes` proprietà dello stream per determinare se esportare in Amazon S3 come caricamento singolo o multiparte. I componenti Greengrass definiti dall'utente impostano questa soglia quando creano uno stream che esporta in Amazon S3. La soglia predefinita è di 5 MB.

Consulta anche

- Gestisci i flussi di dati sui dispositivi core Greengrass
- Utilizzalo StreamManagerClient per lavorare con gli stream
- Esporta configurazioni per le destinazioni supportate Cloud AWS

Esecuzione dell'inferenza di Machine Learning

ConAWS IoT Greengrass, puoi eseguire inferenze di machine learning (ML) sui tuoi dispositivi edge su dati generati localmente utilizzando modelli addestrati sul cloud. In questo modo si beneficia della bassa latenza e dei risparmi sui costi dell'inferenza locale e si sfrutta la potenza del cloud computing per modelli di formazione ed elaborazioni complesse.

AWS IoT Greengrassrende più efficienti i passaggi necessari per eseguire l'inferenza. È possibile addestrare i modelli di inferenza ovunque e distribuirli localmente come componenti di apprendimento automatico. Ad esempio, puoi creare e addestrare modelli di deep learning in Amazon SageMaker o modelli di visione artificiale in Amazon Lookout for Vision. Quindi, puoi archiviare questi modelli in un bucket Amazon S3, in modo da poterli utilizzare come artefatti nei tuoi componenti per eseguire inferenze sui tuoi dispositivi principali.

Argomenti

Come funziona un'inferenza di Machine Learning di AWS IoT Greengrass

AWSfornisce componenti di machine learning che puoi utilizzare per creare distribuzioni in un unico passaggio per eseguire inferenze di machine learning sul tuo dispositivo. È inoltre possibile utilizzare questi componenti come modelli per creare componenti personalizzati in grado di soddisfare requisiti specifici.

AWSfornisce le seguenti categorie di componenti di machine learning:

- Componente del modello: contiene modelli di apprendimento automatico come artefatti Greengrass.
- Componente Runtime: contiene lo script che installa il framework di machine learning e le sue dipendenze sul dispositivo principale Greengrass.
- Componente di inferenza: contiene il codice di inferenza e include le dipendenze dei componenti per installare il framework di machine learning e scaricare modelli di machine learning preaddestrati.

Ogni implementazione creata per eseguire l'inferenza dell'apprendimento automatico è costituita da almeno un componente che esegue l'applicazione di inferenza, installa il framework di apprendimento automatico e scarica i modelli di apprendimento automatico. Per eseguire l'inferenza di esempio con i componenti AWS forniti, distribuisci un componente di inferenza sul tuo dispositivo principale, che include automaticamente il modello e i componenti di runtime corrispondenti come dipendenze. Per personalizzare le distribuzioni, è possibile collegare o sostituire i componenti del modello di esempio con componenti del modello personalizzati oppure utilizzare le ricette dei componenti AWS forniti come modelli per creare componenti di inferenza, modello e runtime personalizzati.

Per eseguire inferenze di machine learning utilizzando componenti personalizzati:

1. Crea un componente del modello. Questo componente contiene i modelli di machine learning da utilizzare per eseguire l'inferenza. AWSfornisce esempi di modelli DLR e TensorFlow Lite pre-addestrati. Per utilizzare un modello personalizzato, crea il tuo componente del modello.
2. Crea un componente runtime. Questo componente contiene gli script necessari per installare il runtime di machine learning per i tuoi modelli. AWSfornisce componenti di runtime di esempio per Deep Learning Runtime (DLR) e TensorFlow Lite. Per utilizzare altri runtime con i tuoi modelli personalizzati e il codice di inferenza, crea i tuoi componenti di runtime.
3. Crea un componente di inferenza. Questo componente contiene il codice di inferenza e include i componenti del modello e del runtime come dipendenze. AWSfornisce componenti di inferenza di esempio per la classificazione delle immagini e il rilevamento di oggetti tramite DLR e Lite. TensorFlow Per eseguire altri tipi di inferenza o utilizzare modelli e runtime personalizzati, create il vostro componente di inferenza.
4. Implementa il componente di inferenza. Quando distribuisci questo componente, distribuisce automaticamente AWS IoT Greengrass anche le dipendenze del modello e del componente di runtime.

Per iniziare con i componenti AWS forniti, consulta. the section called “Esegui un'inferenza di classificazione delle immagini di esempio”

Per informazioni sulla creazione di componenti di machine learning personalizzati, consultaPersonalizza i tuoi componenti di machine learning.

Cosa c'è di diverso nella AWS IoT Greengrass versione 2?

AWS IoT Greengrassconsolida le unità funzionali per l'apprendimento automatico, come modelli, runtime e codice di inferenza, in componenti che consentono di utilizzare un processo in un'unica fase per installare il runtime di machine learning, scaricare i modelli addestrati ed eseguire inferenze sul dispositivo.

Utilizzando i componenti di machine learning AWS forniti, hai la flessibilità di iniziare a eseguire inferenze di machine learning con codice di inferenza di esempio e modelli preaddestrati. È possibile collegare componenti di modelli personalizzati per utilizzare modelli personalizzati con i componenti di inferenza e di runtime forniti. AWS Per una soluzione di machine learning completamente personalizzata, è possibile utilizzare i componenti pubblici come modelli per creare componenti personalizzati e utilizzare qualsiasi tipo di runtime, modello o inferenza desiderato.

Requisiti

Per creare e utilizzare componenti di machine learning, è necessario disporre di quanto segue:

- Un dispositivo principale Greengrass. Se non lo hai, consultare Tutorial: Nozioni di base su AWS IoT Greengrass V2.
- Almeno 500 MB di spazio di archiviazione locale per utilizzare i componenti AWS di machine learning di esempio forniti.

Origini di modello supportate

AWS IoT Greengrasssupporta l'utilizzo di modelli di machine learning personalizzati archiviati in Amazon S3. Puoi anche utilizzare Amazon SageMaker Edge Packaging Job per creare direttamente componenti del modello per i tuoi modelli SageMaker NEO-compilati. Per informazioni sull'utilizzo di SageMaker Edge Manager conAWS IoT Greengrass, consulta. Usa Amazon SageMaker Edge Manager sui dispositivi core Greengrass Puoi anche utilizzare i processi di packaging dei modelli Amazon Lookout for Vision per creare componenti del modello per i tuoi modelli Lookout for Vision.

Per ulteriori informazioni sull'utilizzo di Lookout for Vision AWS IoT Greengrass con, Usa Amazon Lookout Greengrass visione. vedere.

I bucket S3 che contengono i tuoi modelli devono soddisfare i seguenti requisiti:

- Non devono essere crittografati utilizzando SSE-C. Per i bucket che utilizzano la crittografia lato server, l'inferenza dell'apprendimento AWS IoT Greengrass automatico attualmente supporta solo le opzioni di crittografia SSE-S3 o SSE-KMS. Per ulteriori informazioni sulle opzioni di crittografia lato server, consulta Protezione dei dati utilizzando la crittografia lato server nella Guida per l'utente di Amazon Simple Storage Service.
- I loro nomi non devono includere periodi (). `.` Per ulteriori informazioni, consulta la regola sull'utilizzo di bucket in stile host virtuale con SSL nelle Regole per la denominazione dei bucket nella Amazon Simple Storage Service User Guide.
- I bucket S3 che memorizzano i sorgenti del modello devono trovarsi negli stessi componenti di machine learning. Account AWS Regione AWS
- AWS IoT Greengrassdeve avere l'`read`autorizzazione alla fonte del modello. Per consentire l'accesso AWS IoT Greengrass ai bucket S3, il ruolo del dispositivo Greengrass deve consentire l'azione. `s3:GetObject` Per ulteriori informazioni sul ruolo del dispositivo, consulta. Autorizza i dispositivi principali a interagire conAWSservizi

Runtime di machine learning supportati

AWS IoT Greengrassti consente di creare componenti personalizzati per utilizzare qualsiasi runtime di machine learning di tua scelta per eseguire inferenze di machine learning con i tuoi modelli addestrati su misura. Per informazioni sulla creazione di componenti di machine learning personalizzati, consulta. Personalizza i tuoi componenti di machine learning

Per rendere più efficiente il processo di introduzione all'apprendimento automatico, AWS IoT Greengrass fornisce esempi di componenti di inferenza, modello e runtime che utilizzano i seguenti runtime di machine learning:

- Deep Learning Runtime (DLR) v1.6.0 e v1.3.0
- TensorFlow Lite v2.5.0

AWS-componenti di apprendimento automatico forniti

La tabella seguente elenca i componenti AWS forniti utilizzati per l'apprendimento automatico.

> **Note**
>
> Diversi componenti AWS forniti dipendono da versioni minori specifiche del nucleo Greengrass. A causa di questa dipendenza, è necessario aggiornare questi componenti quando si aggiorna il nucleo di Greengrass a una nuova versione secondaria. Per informazioni sulle versioni specifiche del nucleo da cui dipende ogni componente, consultate l'argomento relativo ai componenti. Per ulteriori informazioni sull'aggiornamento del nucleo, vedere. Aggiornamento del software AWS IoT Greengrass Core (OTA)

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Lookout for Vision Edge Agent	Implementa il runtime Amazon Lookout for Vision sul dispositivo principale Greengrass, in modo da poter utilizzare la visione artificiale per trovare difetti nei prodotti industriali.	Generico	Linux	No
SageMaker Edge Manager	Implementa l'agente Amazon SageMaker Edge Manager sul dispositi	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
	vo principale Greengrass.			
Classificazione delle immagini DLR	Componente di inferenza che utilizza l'archivio del modello di classificazione delle immagini DLR e il componente runtime DLR come dipendenze per installare DLR, scaricare modelli di classificazione delle immagini di esempio ed eseguire inferenze di classificazione delle immagini sui dispositivi supportati.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Rilevamento di oggetti DLR	Componente di inferenza che utilizza l'archivio del modello di rilevamento degli oggetti DLR e il componente runtime DLR come dipendenze per installare DLR, scaricare modelli di rilevamento di oggetti di esempio ed eseguire inferenze per il rilevamento degli oggetti sui dispositivi supportati.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Archivio modelli di classificazione delle immagini DLR	Componente del modello che contiene esempi di modelli di classificazione delle immagini ResNet -50 come artefatti Greengrass.	Generico	Linux, Windows	No
Archivio modelli DLR per il rilevamento di oggetti	Componente del modello che contiene esempi di modelli di rilevamento di oggetti YOLov3 come artefatti Greengrass.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
Runtime DLR	Componente di runtime che contiene uno script di installazione utilizzato per installare DLR e le sue dipendenze sul dispositivo principale Greengrass.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
TensorFlow Classificazione delle immagini Lite	Componente di inferenza che utilizza l' TensorFlow archivio del modello di classificazione delle immagini TensorFlow Lite e il componente runtime Lite come dipendenze per installare TensorFlow Lite, scaricare modelli di classificazione delle immagini di esempio ed eseguire inferenze di classificazione delle immagini sui dispositivi supportati.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
TensorFlow Rilevamento di oggetti Lite	Componente di inferenza che utilizza l'archivio del modello di rilevamento degli oggetti TensorFlow Lite e il componente runtime TensorFlow Lite come dipendenze per installare TensorFlow Lite, scaricare modelli di rilevamento di oggetti di esempio ed eseguire inferenze per il rilevamento degli oggetti sui dispositivi supportati.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
TensorFlow Archivio di modelli di classificazione delle immagini Lite	Componente del modello che contiene un modello MobileNet v1 di esempio come artefatto Greengrass.	Generico	Linux, Windows	No
TensorFlow Archivio modelli Lite per il rilevamento di oggetti	Componente del modello che contiene un MobileNet modello di Single Shot Detection (SSD) di esempio come artefatto Greengrass.	Generico	Linux, Windows	No

Componente	Descrizione	Tipo di componente	Sistema operativo supportato	Open source
TensorFlow Runtime Lite	Componente di runtime che contiene uno script di installazione utilizzato per installare TensorFlow Lite e le sue dipendenze sul dispositivo principale Greengrass.	Generico	Linux, Windows	No

Usa Amazon SageMaker Edge Manager sui dispositivi core Greengrass

> **Important**
>
> SageMaker Edge Manager verrà interrotto il 26 aprile 2024. Per ulteriori informazioni su come continuare a distribuire i modelli sui dispositivi edge, consulta SageMaker Edge Manager End of Life.

Amazon SageMaker Edge Manager è un agente software che funziona su dispositivi edge. SageMaker Edge Manager fornisce la gestione dei modelli per i dispositivi edge in modo da poter impacchettare e utilizzare i modelli SageMaker compilati da Amazon NEO direttamente sui dispositivi core Greengrass. Utilizzando SageMaker Edge Manager, puoi anche campionare i dati di input e output del modello dai tuoi dispositivi principali e inviarli a loro Cloud AWS per il monitoraggio e l'analisi. Poiché SageMaker Edge Manager utilizza SageMaker Neo per ottimizzare i modelli per l'hardware di destinazione, non è necessario installare il runtime DLR direttamente sul dispositivo.

Sui dispositivi Greengrass, SageMaker Edge Manager non carica AWS IoT certificati locali né chiama direttamente l'endpoint del fornitore di AWS IoT credenziali. Invece, SageMaker Edge Manager utilizza il servizio di scambio di token per recuperare credenziali temporanee da un endpoint TES.

Questa sezione descrive come funziona SageMaker Edge Manager sui dispositivi core Greengrass.

Come funziona SageMaker Edge Manager sui dispositivi Greengrass

Per distribuire l'agente SageMaker Edge Manager sui tuoi dispositivi principali, crea una distribuzione che includa il `aws.greengrass.SageMakerEdgeManager` componente. AWS IoT Greengrassgestisce l'installazione e il ciclo di vita dell'agente Edge Manager sui dispositivi. Quando è disponibile una nuova versione dell'agente binario, distribuisci la versione aggiornata del `aws.greengrass.SageMakerEdgeManager` componente per aggiornare la versione dell'agente installata sul tuo dispositivo.

Quando si utilizza SageMaker Edge Manager conAWS IoT Greengrass, il flusso di lavoro include i seguenti passaggi di alto livello:

1. Compila modelli con SageMaker Neo.
2. Package dei modelli SageMaker NEO-compilati utilizzando i processi di SageMaker edge packaging. Quando esegui un processo di edge packaging per il tuo modello, puoi scegliere di creare un componente del modello con il modello confezionato come artefatto che può essere distribuito sul tuo dispositivo principale Greengrass.
3. Crea un componente di inferenza personalizzato. Questo componente di inferenza viene utilizzato per interagire con l'agente Edge Manager per eseguire l'inferenza sul dispositivo principale. Queste operazioni includono il caricamento dei modelli, l'invocazione di richieste di previsione per eseguire l'inferenza e lo scaricamento dei modelli quando il componente si spegne.
4. Implementa il componente SageMaker Edge Manager, il componente del modello in pacchetto e il componente di inferenza per eseguire il modello sul motore di SageMaker inferenza (agente Edge Manager) sul dispositivo.

Per ulteriori informazioni sulla creazione di processi di edge packaging e componenti di inferenza compatibili con SageMaker Edge Manager, consulta Deploy Model Package and Edge Manager Agent with AWS IoT Greengrass nella Amazon SageMaker Developer Guide.

Il Tutorial: Inizia a usare SageMaker Edge Manager tutorial mostra come configurare e utilizzare l'agente SageMaker Edge Manager su un dispositivo principale Greengrass esistente, utilizzando

il codice di esempio AWS fornito che è possibile utilizzare per creare componenti di inferenza e modello di esempio.

Quando si utilizza SageMaker Edge Manager sui dispositivi principali Greengrass, è anche possibile utilizzare la funzione di acquisizione dati per caricare dati di esempio su. Cloud AWS L'acquisizione dei dati è una SageMaker funzionalità che utilizzi per caricare input di inferenza, risultati di inferenza e dati di inferenza aggiuntivi su un bucket S3 o una directory locale per analisi future. Per ulteriori informazioni sull'utilizzo dei dati di acquisizione con SageMaker Edge Manager, consulta Manage Model nella Amazon SageMaker Developer Guide.

Requisiti

È necessario soddisfare i seguenti requisiti per utilizzare l'agente SageMaker Edge Manager sui dispositivi principali Greengrass.

- Un dispositivo core Greengrass in esecuzione su Amazon Linux 2, una piattaforma Linux basata su Debian (x86_64 o Armv8) o Windows (x86_64). Se non lo hai, consultare Tutorial: Nozioni di base su AWS IoT Greengrass V2.
- Python 3.6 o versione successiva, inclusa `pip` la tua versione di Python, installato sul tuo dispositivo principale.
- Il ruolo del dispositivo Greengrass è configurato con quanto segue:
  - Una relazione di fiducia che consente `credentials.iot.amazonaws.com` e consente `sagemaker.amazonaws.com` di assumere il ruolo, come illustrato nel seguente esempio di policy IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.iot.amazonaws.com"
       },
      "Action": "sts:AssumeRole"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sagemaker.amazonaws.com"
      },
```

```
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- La politica gestita da AmazonSageMakerEdgeDeviceFleetPolicyIAM.
- L'`s3:PutObject`azione, come illustrato nel seguente esempio di policy IAM.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "*"
      ],
      "Effect": "Allow"
    }
  ]
}
```

- Un bucket Amazon S3 creato nello stesso Regione AWS dispositivo Account AWS principale Greengrass. SageMaker Edge Manager richiede un bucket S3 per creare una flotta di dispositivi edge e per archiviare dati di esempio derivanti dall'esecuzione dell'inferenza sul dispositivo. Per informazioni sulla creazione di bucket S3, consulta Guida introduttiva ad Amazon S3.
- Una flotta di dispositivi SageMaker edge che utilizza lo stesso alias di AWS IoT ruolo del dispositivo principale Greengrass. Per ulteriori informazioni, consulta Crea una flotta di dispositivi edge.
- Il tuo dispositivo principale Greengrass è stato registrato come dispositivo edge nel tuo parco dispositivi SageMaker Edge. Il nome del dispositivo edge deve corrispondere al nome dell'AWS IoToggetto del dispositivo principale. Per ulteriori informazioni, consulta Registra il tuo dispositivo Greengrass core.

Inizia a usare SageMaker Edge Manager

Puoi completare un tutorial per iniziare a usare SageMaker Edge Manager. Il tutorial mostra come iniziare a utilizzare SageMaker Edge Manager con componenti AWS di esempio forniti su un dispositivo principale esistente. Questi componenti di esempio utilizzano il componente SageMaker

Edge Manager come dipendenza per distribuire l'agente Edge Manager ed eseguono l'inferenza utilizzando modelli preaddestrati compilati utilizzando Neo. SageMaker Per ulteriori informazioni, consulta Tutorial: Inizia a usare SageMaker Edge Manager.

Usa Amazon Lookout Greengrass visione.

> **Note**
>
> AWS IoT Greengrassattualmente non supporta questa funzionalità sui dispositivi Windows core.

Amazon Lookout for Vision è uno strumentoServizio AWS che puoi utilizzare per trovare difetti visivi nei prodotti industriali. La visione è in grado di identificare componenti mancanti in un prodotto industriale, danni a veicoli o strutture, irregolarità nelle linee di produzione, condensatori mancanti nei sottili dischi di silicio o qualsiasi altro elemento fisico in cui la qualità è importante. Per ulteriori informazioni, consulta Che cos'è Amazon Lookout for Vision? nella Guida per gli sviluppatori di Amazon Lookout for Vision.

È possibile creare applicazioni Greengrass che utilizzano l'inferenza Lookout for Vision per trovare difetti visivi sui dispositivi principali di Greengrass. Dopo aver distribuito un flusso di lavoro Lookout for Vision su un dispositivo principale di Greengrass, è possibile eseguire la visione artificiale senza una connessione al servizio Lookout for Vision inCloud AWS. Per creare un'applicazione Greengrass che utilizza Lookout for Vision, è necessario configurare e distribuire i seguenti componenti Greengrass:

- Componenti del modello Lookout for Vision: contiene modelli di apprendimento automatico Lookout for Vision come artefatti Greengrass. Puoi utilizzare la console e l'API di Lookout for Vision per generare componenti di modello che raggruppano i modelli di machine learning preaddestrati. Questi componenti sono componenti privati di Greengrass nel tuoAccount AWS. Per ulteriori informazioni, consulta Creazione di un modello Lookout for Vision e Imballaggio di un modello Lookout for Vision nella Amazon Lookout for Vision Developer Guide.
- Componente Lookout for Vision Edge Agent: fornisce un server di runtime locale Lookout for Vision che utilizza la visione artificiale per rilevare le anomalie utilizzando i modelli di apprendimento automatico forniti dall'utente. Questo componente è un componenteAWS fornito. Per ulteriori informazioni, consultare il componente Lookout for Vision.

- Componente dell'applicazione client Lookout for Vision: interagisce con il componente Lookout for Vision Edge Agent per elaborare le immagini alla ricerca di anomalie. È possibile sviluppare componenti di applicazioni client personalizzati che inviano immagini e flussi video all'agente Lookout for Vision Edge locale e segnalano eventuali anomalie rilevate dai modelli di apprendimento automatico. Per ulteriori informazioni, consulta Scrittura di un componente dell'applicazione client e riferimento all'API Lookout for Vision Edge Agent nella Amazon Lookout for Vision Developer Guide.

Per ulteriori informazioni su come creare, configurare e utilizzare questi componenti, consulta Utilizzo di un modello Lookout for Vision su un dispositivo edge nella Amazon Lookout for Vision Developer Guide.

Personalizza i tuoi componenti di machine learning

InAWS IoT Greengrass, puoi configurare componenti di machine learning di esempio per personalizzare il modo in cui esegui l'inferenza di machine learning sui tuoi dispositivi utilizzando i componenti di inferenza, modello e runtime come elementi costitutivi. AWS IoT Greengrassoffre inoltre la flessibilità necessaria per utilizzare i componenti di esempio come modelli e creare componenti personalizzati in base alle esigenze. È possibile combinare questo approccio modulare per personalizzare i componenti di inferenza dell'apprendimento automatico nei seguenti modi:

Utilizzo di componenti di inferenza di esempio

- Modifica la configurazione dei componenti di inferenza quando li distribuisci.
- Utilizzate un modello personalizzato con il componente di inferenza del campione sostituendo il componente sample model store con un componente del modello personalizzato. Il modello personalizzato deve essere addestrato utilizzando lo stesso runtime del modello di esempio.

Utilizzo di componenti di inferenza personalizzati

- Usa codice di inferenza personalizzato con i modelli e i runtime di esempio aggiungendo componenti del modello pubblico e componenti di runtime come dipendenze dei componenti di inferenza personalizzati.
- Crea e aggiungi componenti di modello personalizzati o componenti di runtime come dipendenze di componenti di inferenza personalizzati. È necessario utilizzare componenti personalizzati se si desidera utilizzare un codice di inferenza personalizzato o un runtime per il quale AWS IoT Greengrass non è disponibile un componente di esempio.

Argomenti

Modifica la configurazione di un componente di inferenza pubblica

Nella AWS IoT Greengrassconsole, la pagina del componente mostra la configurazione predefinita di quel componente. Ad esempio, la configurazione predefinita del componente di classificazione delle immagini TensorFlow Lite è simile alla seguente:

```
{
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.TensorFlowLiteImageClassification:mqttproxy:1": {
        "policyDescription": "Allows access to publish via topic ml/tflite/image-
classification.",
        "operations": [
          "aws.greengrass#PublishToIoTCore"
        ],
        "resources": [
          "ml/tflite/image-classification"
        ]
      }
    }
  },
  "PublishResultsOnTopic": "ml/tflite/image-classification",
  "ImageName": "cat.jpeg",
  "InferenceInterval": 3600,
  "ModelResourceKey": {
    "model": "TensorFlowLite-Mobilenet"
  }
}
```

Quando si distribuisce un componente di inferenza pubblica, è possibile modificare la configurazione predefinita per personalizzare la distribuzione. Per informazioni sui parametri di configurazione disponibili per ogni componente di inferenza pubblica, consulta l'argomento relativo al componente in. AWS-componenti di apprendimento automatico forniti

Questa sezione descrive come distribuire un componente modificato dalla AWS IoT Greengrass console. Per informazioni sulla distribuzione dei componenti utilizzando ilAWS CLI, vedere. Creare distribuzione

Per distribuire un componente di inferenza pubblica modificato (console)

1. Accedi alla console AWS IoT Greengrass.
2. Nel menu di navigazione, scegli Componenti.
3. Nella pagina Componenti, nella scheda Componenti pubblici, scegli il componente che desideri distribuire.
4. Nella pagina del componente, scegli Distribuisci.
5. Da Aggiungi alla distribuzione, scegli una delle seguenti opzioni:
   a. Per unire questo componente a una distribuzione esistente sul dispositivo di destinazione, scegli Aggiungi alla distribuzione esistente, quindi seleziona la distribuzione che desideri modificare.
   b. Per creare una nuova distribuzione sul dispositivo di destinazione, scegli Crea nuova distribuzione. Se hai una distribuzione esistente sul tuo dispositivo, la scelta di questo passaggio sostituisce la distribuzione esistente.
6. Nella pagina Specifica destinazione, procedi come segue:
   a. In Informazioni sulla distribuzione, inserisci o modifica il nome descrittivo della distribuzione.
   b. In Destinazione della distribuzione, seleziona una destinazione della distribuzione e scegli Avanti. Non è possibile modificare la destinazione della distribuzione se si sta revisionando una distribuzione esistente.
7. Nella pagina Seleziona componenti, in Componenti pubblici, verifica che sia selezionato il componente di inferenza con la configurazione modificata e scegli Avanti.
8. Nella pagina Configura componenti, procedi come segue:
   a. Selezionate il componente di inferenza e scegliete Configura componente.
   b. In Aggiornamento della configurazione, inserisci i valori di configurazione che desideri aggiornare. Ad esempio, inserisci il seguente aggiornamento di configurazione nella casella Configurazione da unire per modificare l'intervallo di inferenza a 15 secondi e chiedi al componente di cercare l'immagine denominata `custom.jpg` nella cartella. `/custom-ml-inference/images/`

```
{
  "InferenceInterval": "15",
  "ImageName": "custom.jpg",
  "ImageDirectory": "/custom-ml-inference/images/"
}
```

Per ripristinare l'intera configurazione di un componente ai valori predefiniti, specificate una singola stringa vuota **""** nella casella Reimposta percorsi.

c. Seleziona Conferma e scegli Avanti.

9. Nella pagina Configura impostazioni avanzate, mantieni le impostazioni di configurazione predefinite e scegli Avanti.
10. Nella pagina di revisione, scegli Distribuisci

Utilizza un modello personalizzato con il componente di inferenza del campione

Se desideri utilizzare il componente di inferenza del campione con i tuoi modelli di machine learning per un runtime che AWS IoT Greengrass fornisce un componente di runtime di esempio, devi sostituire i componenti del modello pubblico con componenti che utilizzano tali modelli come artefatti. A un livello elevato, completate i seguenti passaggi per utilizzare un modello personalizzato con il componente di inferenza di esempio:

1. Crea un componente del modello che utilizza un modello personalizzato in un bucket S3 come artefatto. Il modello personalizzato deve essere addestrato utilizzando lo stesso runtime del modello che si desidera sostituire.
2. Modifica il parametro `ModelResourceKey` di configurazione nel componente di inferenza per utilizzare il modello personalizzato. Per informazioni sull'aggiornamento della configurazione del componente di inferenza, vedere Modifica la configurazione di un componente di inferenza pubblica

Quando si distribuisce il componente di inferenza, AWS IoT Greengrass cerca la versione più recente delle dipendenze dei componenti. Sostituisce il componente del modello pubblico dipendente se nello stesso e è presente una versione personalizzata successiva del componente. Account AWS Regione AWS

Crea un componente del modello personalizzato (console)

1. Carica il tuo modello in un bucket S3. Per informazioni sul caricamento dei modelli in un bucket S3, consulta Working with Amazon S3 Buckets nella Amazon Simple Storage Service User Guide.

   > **Note**
   >
   > È necessario archiviare gli artefatti in bucket S3 che si trovano negli stessi componenti. Account AWS Regione AWS Per consentire l'accesso AWS IoT Greengrass a questi artefatti, il ruolo del dispositivo Greengrass deve consentire l'azione. `s3:GetObject` Per ulteriori informazioni sul ruolo del dispositivo, vedere. Autorizza i dispositivi principali a interagire conAWSservizi

2. Nel menu di navigazione della AWS IoT Greengrassconsole, scegli Componenti.
3. Recupera la ricetta del componente Public Model Store.

   a. Nella pagina Componenti, nella scheda Componenti pubblici, cercate e scegliete il componente del modello pubblico per il quale desiderate creare una nuova versione. Ad esempio, `variant.DLR.ImageClassification.ModelStore`.

   b. Nella pagina del componente, scegli Visualizza ricetta e copia la ricetta JSON visualizzata.
4. Nella pagina Componenti, nella scheda I miei componenti, scegli Crea componente.
5. Nella pagina Crea componente, in Informazioni sui componenti, seleziona Inserisci la ricetta come JSON come origine del componente.
6. Nella casella Ricetta, incolla la ricetta del componente che hai copiato in precedenza.
7. Nella ricetta, aggiorna i seguenti valori:

   - `ComponentVersion`: Incrementa la versione secondaria del componente.

     Quando create un componente personalizzato per sostituire un componente del modello pubblico, dovete aggiornare solo la versione secondaria della versione del componente esistente. Ad esempio, se la versione del componente pubblico è`2.1.0`, puoi creare un componente personalizzato con version`2.1.1`.
   - `Manifests.Artifacts.Uri`: aggiorna ogni valore URI all'URI Amazon S3 del modello che desideri utilizzare.

> **Note**
>
> Non modificare il nome del componente.

8. Scegliete Crea componente.

Crea un componente del modello personalizzato (AWS CLI)

1. Carica il tuo modello in un bucket S3. Per informazioni sul caricamento dei modelli in un bucket S3, consulta Working with Amazon S3 Buckets nella Amazon Simple Storage Service User Guide.

   > **Note**
   >
   > È necessario archiviare gli artefatti in bucket S3 che si trovano negli stessi componenti. Account AWS Regione AWS Per consentire l'accesso AWS IoT Greengrass a questi artefatti, il ruolo del dispositivo Greengrass deve consentire l'azione. `s3:GetObject` Per ulteriori informazioni sul ruolo del dispositivo, vedere. Autorizza i dispositivi principali a interagire conAWSservizi

2. Eseguite il comando seguente per recuperare la ricetta del componente pubblico. Questo comando scrive la ricetta del componente nel file di output fornito nel comando. Convertite la stringa recuperata con codifica base64 in JSON o YAML, se necessario.

   Linux, macOS, or Unix

   ```
   aws greengrassv2 get-component \
       --arn <arn> \
       --recipe-output-format <recipe-format> \
       --query recipe \
       --output text | base64 --decode > <recipe-file>
   ```

   Windows Command Prompt (CMD)

   ```
   aws greengrassv2 get-component ^
       --arn <arn> ^
       --recipe-output-format <recipe-format> ^
       --query recipe ^
   ```

```
    --output text > <recipe-file>.base64

certutil -decode <recipe-file>.base64 <recipe-file>
```

PowerShell

```
aws greengrassv2 get-component `
    --arn <arn> `
    --recipe-output-format <recipe-format> `
    --query recipe `
    --output text > <recipe-file>.base64

certutil -decode <recipe-file>.base64 <recipe-file>
```

3. Aggiorna il nome del file di ricetta in*<component-name>*-*<component-version>*, dove la versione del componente è la versione di destinazione del nuovo componente. Ad esempio, `variant.DLR.ImageClassification.ModelStore-2.1.1.yaml`.
4. Nella ricetta, aggiorna i seguenti valori:

   - `ComponentVersion`: Incrementa la versione secondaria del componente.

     Quando create un componente personalizzato per sostituire un componente del modello pubblico, dovete aggiornare solo la versione secondaria della versione del componente esistente. Ad esempio, se la versione del componente pubblico è`2.1.0`, puoi creare un componente personalizzato con version`2.1.1`.

   - `Manifests.Artifacts.Uri`: aggiorna ogni valore URI all'URI Amazon S3 del modello che desideri utilizzare.

   > **Note**
   >
   > Non modificare il nome del componente.

5. Esegui il seguente comando per creare un nuovo componente utilizzando la ricetta che hai recuperato e modificato.

```
aws greengrassv2 create-component-version \
    --inline-recipe fileb://path/to/component/recipe
```

> **Note**
>
> Questo passaggio crea il componente nel AWS IoT Greengrass servizio in. Cloud AWS Puoi utilizzare la Greengrass CLI per sviluppare, testare e distribuire il componente localmente prima di caricarlo sul cloud. Per ulteriori informazioni, consulta Sviluppa AWS IoT Greengrass componenti.

Per ulteriori informazioni sulla creazione di componenti, vedere. Sviluppa AWS IoT Greengrass componenti

Crea componenti di machine learning personalizzati

È necessario creare componenti personalizzati se si desidera utilizzare un codice di inferenza personalizzato o un runtime per il quale AWS IoT Greengrass non è disponibile un componente di esempio. È possibile utilizzare il codice di inferenza personalizzato con i modelli e i runtime di apprendimento automatico di esempio AWS forniti oppure sviluppare una soluzione di inferenza di machine learning completamente personalizzata con modelli e runtime propri. Se i modelli utilizzano un runtime per il quale AWS IoT Greengrass è disponibile un componente di runtime di esempio, è possibile utilizzare tale componente di runtime e creare componenti personalizzati solo per il codice di inferenza e i modelli che si desidera utilizzare.

Argomenti

- Recuperate la ricetta per un componente pubblico
- Recupera gli artefatti dei componenti di esempio
- Carica gli artefatti dei componenti in un bucket S3
- Creare componenti personalizzati

Recuperate la ricetta per un componente pubblico

È possibile utilizzare la ricetta di un componente pubblico di apprendimento automatico esistente come modello per creare un componente personalizzato. Per visualizzare la ricetta del componente per la versione più recente di un componente pubblico, usa la console o AWS CLI quanto segue:

- Utilizzo della console

1. Nella pagina Componenti, nella scheda Componenti pubblici, cerca e scegli il componente pubblico.
2. Nella pagina del componente, scegli Visualizza ricetta.

- Uso di AWS CLI

  Esegui il comando seguente per recuperare la ricetta del componente della variante pubblica. Questo comando scrive la ricetta del componente nel file di ricetta JSON o YAML fornito nel comando.

  Linux, macOS, or Unix

  ```
  aws greengrassv2 get-component \
      --arn <arn> \
      --recipe-output-format <recipe-format> \
      --query recipe \
      --output text | base64 --decode > <recipe-file>
  ```

  Windows Command Prompt (CMD)

  ```
  aws greengrassv2 get-component ^
      --arn <arn> ^
      --recipe-output-format <recipe-format> ^
      --query recipe ^
      --output text > <recipe-file>.base64

  certutil -decode <recipe-file>.base64 <recipe-file>
  ```

  PowerShell

  ```
  aws greengrassv2 get-component `
      --arn <arn> `
      --recipe-output-format <recipe-format> `
      --query recipe `
      --output text > <recipe-file>.base64

  certutil -decode <recipe-file>.base64 <recipe-file>
  ```

  Sostituisci i valori nel tuo comando come segue:

  - *<arn>*. L'Amazon Resource Name (ARN) del componente pubblico.

- *<recipe-format>*. Il formato in cui si desidera creare il file delle ricette. I valori supportati sono `JSON` e `YAML`.
- *<recipe-file>*. Il nome della ricetta nel formato*<component-name>*-*<component-version>*.

Recupera gli artefatti dei componenti di esempio

È possibile utilizzare gli artefatti utilizzati dai componenti pubblici di machine learning come modelli per creare artefatti di componenti personalizzati, come codice di inferenza o script di installazione in fase di esecuzione.

Per visualizzare gli artefatti di esempio inclusi nei componenti pubblici di machine learning, distribuisci il componente di inferenza pubblica e quindi visualizza gli artefatti sul tuo dispositivo nella cartella. */greengrass/v2*`/packages/artifacts-unarchived/`*component-name*`/`*component-version*`/`

Carica gli artefatti dei componenti in un bucket S3

Prima di poter creare un componente personalizzato, devi caricare gli artefatti del componente in un bucket S3 e utilizzare gli URI S3 nella ricetta del componente. Ad esempio, per utilizzare il codice di inferenza personalizzato nel componente di inferenza, carica il codice in un bucket S3. Puoi quindi utilizzare l'URI Amazon S3 del tuo codice di inferenza come artefatto nel tuo componente.

Per informazioni sul caricamento di contenuti in un bucket S3, consulta Working with Amazon S3 Bucket nella Amazon Simple Storage Service User Guide.

> **Note**
>
> È necessario archiviare gli artefatti in bucket S3 che si trovano negli stessi componenti. Account AWS Regione AWS Per consentire l'accesso AWS IoT Greengrass a questi artefatti, il ruolo del dispositivo Greengrass deve consentire l'azione. `s3:GetObject` Per ulteriori informazioni sul ruolo del dispositivo, vedere. Autorizza i dispositivi principali a interagire conAWSservizi

Creare componenti personalizzati

Puoi utilizzare gli artefatti e le ricette recuperati per creare componenti di machine learning personalizzati. Per vedere un esempio, consulta Crea un componente di inferenza personalizzato.

Per informazioni dettagliate sulla creazione e la distribuzione di componenti sui dispositivi Greengrass, Sviluppa AWS IoT Greengrass componenti vedere e. Implementazione AWS IoT Greengrass dei componenti sui dispositivi

Crea un componente di inferenza personalizzato

Questa sezione mostra come creare un componente di inferenza personalizzato utilizzando il componente di classificazione delle immagini DLR come modello.

Argomenti

- Carica il tuo codice di inferenza in un bucket Amazon S3
- Crea una ricetta per il tuo componente di inferenza
- Creare il componente di inferenza

Carica il tuo codice di inferenza in un bucket Amazon S3

Crea il tuo codice di inferenza e poi caricalo in un bucket S3. Per informazioni sul caricamento di contenuti in un bucket S3, consulta Working with Amazon S3 Bucket nella Amazon Simple Storage Service User Guide.

> **Note**
>
> È necessario archiviare gli artefatti in bucket S3 che si trovano negli stessi componenti. Account AWS Regione AWS Per consentire l'accesso AWS IoT Greengrass a questi artefatti, il ruolo del dispositivo Greengrass deve consentire l'azione. `s3:GetObject` Per ulteriori informazioni sul ruolo del dispositivo, vedere. Autorizza i dispositivi principali a interagire conAWSservizi

Crea una ricetta per il tuo componente di inferenza

1. Eseguite il comando seguente per recuperare la ricetta del componente di classificazione delle immagini DLR. Questo comando scrive la ricetta del componente nel file di ricetta JSON o YAML fornito nel comando.

   Linux, macOS, or Unix

   ```
   aws greengrassv2 get-component \
   ```

```
    --arn
 arn:aws:greengrass:region:aws:components:aws.greengrass.DLRImageClassification:versions
 \
    --recipe-output-format JSON | YAML \
    --query recipe \
    --output text | base64 --decode > <recipe-file>
```

Windows Command Prompt (CMD)

```
aws greengrassv2 get-component ^
    --arn
 arn:aws:greengrass:region:aws:components:aws.greengrass.DLRImageClassification:versions
 ^
    --recipe-output-format JSON | YAML ^
    --query recipe ^
    --output text > <recipe-file>.base64

certutil -decode <recipe-file>.base64 <recipe-file>
```

PowerShell

```
aws greengrassv2 get-component `
    --arn
 arn:aws:greengrass:region:aws:components:aws.greengrass.DLRImageClassification:versions
 `
    --recipe-output-format JSON | YAML `
    --query recipe `
    --output text > <recipe-file>.base64

certutil -decode <recipe-file>.base64 <recipe-file>
```

Sostituisci <recipe-file>con il nome della ricetta nel formato. *<component-name>*-*<component-version>*

2. `ComponentDependencies`Nell'oggetto della ricetta, effettuate una o più delle seguenti operazioni a seconda del modello e dei componenti di runtime che desiderate utilizzare:

   - Mantieni la dipendenza dai componenti DLR se desideri utilizzare modelli compilati con DLR. Potete anche sostituirlo con una dipendenza da un componente di runtime personalizzato, come illustrato nell'esempio seguente.

Componente di runtime

JSON

```
{
    "<runtime-component>": {
        "VersionRequirement": "<version>",
        "DependencyType": "HARD"
    }
}
```

YAML

```
<runtime-component>:
    VersionRequirement: "<version>"
    DependencyType: HARD
```

- Mantieni la dipendenza dall'archivio del modello di classificazione delle immagini DLR per utilizzare i modelli ResNet -50 preaddestrati che AWS fornisce o modificalo per utilizzare un componente del modello personalizzato. Quando includete una dipendenza per un componente del modello pubblico, se nello stesso Account AWS componente esiste una versione personalizzata successivaRegione AWS, il componente di inferenza utilizza quel componente personalizzato. Specificate la dipendenza del componente del modello come illustrato negli esempi seguenti.

  Componente del modello pubblico

  JSON

  ```
  {
      "variant.DLR.ImageClassification.ModelStore": {
          "VersionRequirement": "<version>",
          "DependencyType": "HARD"
      }
  }
  ```

  YAML

  ```
  variant.DLR.ImageClassification.ModelStore:
      VersionRequirement: "<version>"
      DependencyType: HARD
  ```

Componente del modello personalizzato

JSON

```
{
    "<custom-model-component>": {
        "VersionRequirement": "<version>",
        "DependencyType": "HARD"
    }
}
```

YAML

```
<custom-model-component>:
    VersionRequirement: "<version>"
    DependencyType: HARD
```

3. Nell'`ComponentConfiguration`oggetto, aggiungete la configurazione predefinita per questo componente. È possibile modificare questa configurazione in un secondo momento quando si distribuisce il componente. Il seguente estratto mostra la configurazione dei componenti per il componente di classificazione delle immagini DLR.

   Ad esempio, se utilizzate un componente del modello personalizzato come dipendenza per il componente di inferenza personalizzato, modificatelo per `ModelResourceKey` fornire i nomi dei modelli che state utilizzando.

   JSON

```
{
  "accessControl": {
    "aws.greengrass.ipc.mqttproxy": {
      "aws.greengrass.ImageClassification:mqttproxy:1": {
        "policyDescription": "Allows access to publish via topic ml/dlr/image-
classification.",
        "operations": [
          "aws.greengrass#PublishToIoTCore"
        ],
        "resources": [
          "ml/dlr/image-classification"
        ]
      }
```

```
    }
  },
  "PublishResultsOnTopic": "ml/dlr/image-classification",
  "ImageName": "cat.jpeg",
  "InferenceInterval": 3600,
  "ModelResourceKey": {
    "armv7l": "DLR-resnet50-armv7l-cpu-ImageClassification",
    "x86_64": "DLR-resnet50-x86_64-cpu-ImageClassification",
    "aarch64": "DLR-resnet50-aarch64-cpu-ImageClassification"
  }
}
```

YAML

```
accessControl:
    aws.greengrass.ipc.mqttproxy:
        'aws.greengrass.ImageClassification:mqttproxy:1':
            policyDescription: 'Allows access to publish via topic ml/dlr/image-
classification.'
            operations:
                - 'aws.greengrass#PublishToIoTCore'
            resources:
                - ml/dlr/image-classification
PublishResultsOnTopic: ml/dlr/image-classification
ImageName: cat.jpeg
InferenceInterval: 3600
ModelResourceKey:
    armv7l: "DLR-resnet50-armv7l-cpu-ImageClassification"
    x86_64: "DLR-resnet50-x86_64-cpu-ImageClassification"
    aarch64: "DLR-resnet50-aarch64-cpu-ImageClassification"
```

4. Nell'`Manifests`oggetto, fornite informazioni sugli artefatti e sulla configurazione di questo componente che vengono utilizzati quando il componente viene distribuito su piattaforme diverse e qualsiasi altra informazione necessaria per eseguire correttamente il componente. Il seguente estratto mostra la configurazione dell'`Manifests`oggetto per la piattaforma Linux nel componente di classificazione delle immagini DLR.

   JSON

   ```
   {
     "Manifests": [
       {
   ```

```
      "Platform": {
        "os": "linux",
        "architecture": "arm"
      },
      "Name": "32-bit armv7l - Linux (raspberry pi)",
      "Artifacts": [
        {
          "URI": "s3://SAMPLE-BUCKET/sample-artifacts-directory/
image_classification.zip",
          "Unarchive": "ZIP"
        }
      ],
      "Lifecycle": {
        "Setenv": {
          "DLR_IC_MODEL_DIR":
 "{variant.DLR.ImageClassification.ModelStore:artifacts:decompressedPath}/
{configuration:/ModelResourceKey/armv7l}",
          "DEFAULT_DLR_IC_IMAGE_DIR": "{artifacts:decompressedPath}/
image_classification/sample_images/"
        },
        "run": {
          "RequiresPrivilege": true,
          "script": ". {variant.DLR:configuration:/MLRootPath}/
greengrass_ml_dlr_venv/bin/activate\npython3 {artifacts:decompressedPath}/
image_classification/inference.py"
        }
      }
    }
  ]
}
```

YAML

```
Manifests:
  - Platform:
      os: linux
      architecture: arm
    Name: 32-bit armv7l - Linux (raspberry pi)
    Artifacts:
      - URI: s3://SAMPLE-BUCKET/sample-artifacts-directory/
image_classification.zip
        Unarchive: ZIP
    Lifecycle:
```

```
      Setenv:
        DLR_IC_MODEL_DIR:
 "{variant.DLR.ImageClassification.ModelStore:artifacts:decompressedPath}/
{configuration:/ModelResourceKey/armv7l}"
        DEFAULT_DLR_IC_IMAGE_DIR: "{artifacts:decompressedPath}/
image_classification/sample_images/"
      run:
        RequiresPrivilege: true
        script: |-
          . {variant.DLR:configuration:/MLRootPath}/greengrass_ml_dlr_venv/bin/
activate
          python3 {artifacts:decompressedPath}/image_classification/inference.py
```

Per informazioni dettagliate sulla creazione delle ricette dei componenti, vedere. AWS IoT Greengrassriferimento alla ricetta del componente

Creare il componente di inferenza

Usa la AWS IoT Greengrass console o il AWS CLI per creare un componente usando la ricetta che hai appena definito. Dopo aver creato il componente, puoi distribuirlo per eseguire inferenze sul tuo dispositivo. Per un esempio di come implementare un componente di inferenza, vedi. Tutorial: eseguire l'inferenza della classificazione delle immagini di esempio utilizzando TensorFlow Lite

Crea un componente di inferenza personalizzato (console)

1. Accedi alla console AWS IoT Greengrass.
2. Nel menu di navigazione, scegli Componenti.
3. Nella pagina Componenti, nella scheda I miei componenti, scegli Crea componente.
4. Nella pagina Crea componente, in Informazioni sui componenti, seleziona Inserisci la ricetta come JSON o Inserisci la ricetta come YAML come origine del componente.
5. Nella casella Ricetta, inserisci la ricetta personalizzata che hai creato.
6. Fate clic su Crea componente.

Crea un componente di inferenza personalizzato () AWS CLI

Esegui il comando seguente per creare un nuovo componente personalizzato utilizzando la ricetta che hai creato.

```
aws greengrassv2 create-component-version \
```

```
    --inline-recipe fileb://path/to/recipe/file
```

> **Note**
>
> Questo passaggio crea il componente nel AWS IoT Greengrass servizio inCloud AWS. Puoi utilizzare la Greengrass CLI per sviluppare, testare e distribuire il componente localmente prima di caricarlo sul cloud. Per ulteriori informazioni, consulta Sviluppa AWS IoT Greengrass componenti.

Risoluzione dei problemi di inferenza dell'apprendimento automatico

Utilizza le informazioni e le soluzioni per la risoluzione dei problemi contenute in questa sezione per risolvere i problemi relativi ai componenti di machine learning. Per i componenti pubblici di inferenza dell'apprendimento automatico, consulta i messaggi di errore nei seguenti log dei componenti:

Linux or Unix

- */greengrass/v2*/logs/aws.greengrass.DLRImageClassification.log
- */greengrass/v2*/logs/aws.greengrass.DLRObjectDetection.log
- */greengrass/v2*/logs/aws.greengrass.TensorFlowLiteImageClassification.log
- */greengrass/v2*/logs/aws.greengrass.TensorFlowLiteObjectDetection.log

Windows

- *C:\greengrass\v2*\logs\aws.greengrass.DLRImageClassification.log
- *C:\greengrass\v2*\logs\aws.greengrass.DLRObjectDetection.log
- *C:\greengrass\v2*\logs\aws.greengrass.TensorFlowLiteImageClassification.log
- *C:\greengrass\v2*\logs\aws.greengrass.TensorFlowLiteObjectDetection.log

Se un componente è installato correttamente, il registro dei componenti contiene la posizione della libreria utilizzata per l'inferenza.

Problemi

Impossibile recuperare la libreria

Il seguente errore si verifica quando lo script di installazione non riesce a scaricare una libreria richiesta durante la distribuzione su un dispositivo Raspberry Pi.

```
Err:2 http://raspbian.raspberrypi.org/raspbian buster/main armhf python3.7-dev armhf
 3.7.3-2+deb10u1
404 Not Found [IP: 93.93.128.193 80]
E: Failed to fetch http://raspbian.raspberrypi.org/raspbian/pool/main/p/python3.7/
libpython3.7-dev_3.7.3-2+deb10u1_armhf.deb 404 Not Found [IP: 93.93.128.193 80]
```

Esegui `sudo apt-get update` e distribuisci nuovamente il componente.

Cannot open shared object file

Potresti visualizzare errori simili ai seguenti quando lo script di installazione non riesce a scaricare una dipendenza richiesta `opencv-python` durante la distribuzione su un dispositivo Raspberry Pi.

```
ImportError: libopenjp2.so.7: cannot open shared object file: No such file or directory
```

Esegui il comando seguente per installare manualmente le dipendenze per: `opencv-python`

```
sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
 libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
```

Error: ModuleNotFoundError: No module named '<library>'

È possibile che questo errore venga visualizzato nei registri dei componenti di runtime ML (`variant.DLR.logovariant.TensorFlowLite.log`) quando la libreria di runtime ML o le relative dipendenze non sono installate correttamente. Questo errore può verificarsi nei seguenti casi:

- Se si utilizza l'`UseInstaller`opzione, abilitata per impostazione predefinita, questo errore indica che il componente di runtime ML non è riuscito a installare il runtime o le sue dipendenze. Esegui questa operazione:
  1. Configurate il componente di runtime ML per disabilitare l'`UseInstaller`opzione.
  2. Installa il runtime ML e le sue dipendenze e rendili disponibili all'utente di sistema che esegue i componenti ML. Per ulteriori informazioni, consulta gli argomenti seguenti:
     - Opzione di runtime DLR UseInstaller
     - TensorFlowOpzione di runtime UseInstaller Lite
- Se non si utilizza l'`UseInstaller`opzione, questo errore indica che il runtime ML o le relative dipendenze non sono installati per l'utente di sistema che esegue i componenti ML. Esegui questa operazione:
  1. Verificate che la libreria sia installata per l'utente di sistema che esegue i componenti ML. Sostituisci *`ggc_user`* con il nome dell'utente di sistema e sostituisci *`tflite_runtime`* con il nome della libreria da controllare.

     Linux or Unix

     ```
     sudo -H -u ggc_user bash -c "python3 -c 'import tflite_runtime'"
     ```

     Windows

     ```
     runas /user:ggc_user "py -3 -c \"import tflite_runtime\""
     ```
  2. Se la libreria non è installata, installala per quell'utente. Sostituisci *`ggc_user`* con il nome dell'utente di sistema e sostituisci *`tflite_runtime`* con il nome della libreria.

     Linux or Unix

     ```
     sudo -H -u ggc_user bash -c "python3 -m pip install --user tflite_runtime"
     ```

Windows

```
runas /user:ggc_user "py -3 -m pip install --user tflite_runtime"
```

Per ulteriori informazioni sulle dipendenze per ogni runtime ML, vedete quanto segue:

- Opzione di runtime DLR UseInstaller
- TensorFlowOpzione di runtime UseInstaller Lite

3. Se il problema persiste, installa la libreria per un altro utente per confermare se il dispositivo è in grado di installare la libreria. L'utente potrebbe essere, ad esempio, il tuo utente, l'utente root o un utente amministratore. Se non riesci a installare correttamente la libreria per nessun utente, il tuo dispositivo potrebbe non supportare la libreria. Consulta la documentazione della libreria per esaminare i requisiti e risolvere i problemi di installazione.

Non viene rilevato alcun dispositivo compatibile con CUDA

È possibile che venga visualizzato il seguente errore quando si utilizza l'accelerazione GPU. Esegui il comando seguente per abilitare l'accesso alla GPU per l'utente Greengrass.

```
sudo usermod -a -G video ggc_user
```

File o directory inesistenti

I seguenti errori indicano che il componente runtime non è stato in grado di configurare correttamente l'ambiente virtuale:

- *MLRootPath*`/greengrass_ml_dlr_conda/bin/conda: No such file or directory`
- *MLRootPath*`/greengrass_ml_dlr_venv/bin/activate: No such file or directory`
- *MLRootPath*`/greengrass_ml_tflite_conda/bin/conda: No such file or directory`
- *MLRootPath*`/greengrass_ml_tflite_venv/bin/activate: No such file or directory`

Controlla i log per assicurarti che tutte le dipendenze di runtime siano state installate correttamente. Per ulteriori informazioni sulle librerie installate dallo script di installazione, consultate i seguenti argomenti:

- Runtime DLR
- TensorFlow Runtime Lite

Per impostazione predefinita, *ML RootPath* è impostato su. */greengrass/v2/work/component-name/greengrass_ml* Per modificare questa posizione, includi il componente Runtime DLR o TensorFlow Runtime Lite runtime direttamente nella distribuzione e specifica un valore modificato per il `MLRootPath` parametro in un aggiornamento di fusione della configurazione. Per ulteriori informazioni sulla configurazione del componente, vedere. Aggiornamento delle configurazioni dei componenti

> **Note**
>
> Per il componente DLR v1.3.x, si imposta il `MLRootPath` parametro nella configurazione del componente di inferenza e il valore predefinito è. `$HOME/greengrass_ml`

RuntimeError: module compiled against API version 0xf but this version of NumPy is <version>

Potresti visualizzare i seguenti errori quando esegui l'inferenza dell'apprendimento automatico su un Raspberry Pi con sistema operativo Raspberry Pi Bullseye.

```
RuntimeError: module compiled against API version 0xf but this version of numpy is 0xd
ImportError: numpy.core.multiarray failed to import
```

Questo errore si verifica perché il sistema operativo Raspberry Pi Bullseye include una versione precedente NumPy a quella richiesta da OpenCV. Per risolvere questo problema, esegui il seguente comando per eseguire l'aggiornamento alla versione NumPy più recente.

```
pip3 install --upgrade numpy
```

picamera.exc.PiCameraError: Camera is not enabled

Potresti visualizzare il seguente errore quando esegui l'inferenza dell'apprendimento automatico su un Raspberry Pi che esegue Raspberry Pi OS Bullseye.

```
picamera.exc.PiCameraError: Camera is not enabled. Try running 'sudo raspi-config' and
 ensure that the camera has been enabled.
```

Questo errore si verifica perché Raspberry Pi OS Bullseye include un nuovo stack di fotocamere che non è compatibile con i componenti ML. Per risolvere questo problema, abilita lo stack di fotocamere precedente.

Per abilitare lo stack di telecamere precedente

1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

   ```
   sudo raspi-config
   ```

2. Seleziona Opzioni di interfaccia.
3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
4. Riavvia il dispositivo Raspberry Pi.

Errori di memoria

I seguenti errori si verificano in genere quando il dispositivo non dispone di memoria sufficiente e il processo del componente viene interrotto.

- `stderr. Killed.`
- `exitCode=137`

Si consiglia un minimo di 500 MB di memoria per distribuire un componente di inferenza pubblico per l'apprendimento automatico.

Errori di spazio su disco

L'`no space left on device`errore si verifica in genere quando un dispositivo non dispone di spazio di archiviazione sufficiente. Assicurati che ci sia abbastanza spazio su disco disponibile sul tuo dispositivo prima di distribuire nuovamente il componente. Consigliamo un minimo di 500 MB di spazio libero su disco per distribuire un componente di inferenza di machine learning pubblico.

Errori di timeout

I componenti di machine learning pubblici scaricano file di modelli di machine learning di grandi dimensioni che superano i 200 MB. Se il download scade durante la distribuzione, controlla la velocità della tua connessione Internet e riprova la distribuzione.

Gestisci i dispositivi core Greengrass con AWS Systems Manager

> **Note**
>
> AWS IoT Greengrassattualmente non supporta questa funzionalità sui dispositivi Windows core.

Systems Manager è un AWS servizio che puoi utilizzare per visualizzare e controllare la tua infrastrutturaAWS, tra cui istanze Amazon EC2, server e macchine virtuali (VM) locali e dispositivi periferici. Systems Manager consente di visualizzare i dati operativi, automatizzare le attività operative e mantenere la sicurezza e la conformità. Quando si registra una macchina con Systems Manager, viene chiamata nodo gestito. Per ulteriori informazioni, consulta Che cos'è AWS Systems Manager? nella Guida per l'utente di AWS Systems Manager.

L'AWS Systems Manageragente (Systems Manager Agent) è un software che è possibile installare sui dispositivi per consentire a Systems Manager di aggiornarli, gestirli e configurarli. Per installare Systems Manager Agent sui dispositivi core Greengrass, distribuisci il componente Systems Manager Agent. Quando si distribuisce Systems Manager Agent per la prima volta, il dispositivo principale viene registrato come nodo gestito di Systems Manager. L'agente Systems Manager viene eseguito sul dispositivo per consentire la comunicazione con il servizio Systems Manager inCloud AWS. Per ulteriori informazioni su come installare e configurare il componente Systems Manager Agent, vedereInstallare l'agente AWS Systems Manager.

Gli strumenti e le funzionalità di Systems Manager sono denominati funzionalità. I dispositivi core Greengrass supportano tutte le funzionalità di Systems Manager. Per ulteriori informazioni su queste funzionalità e su come utilizzare Systems Manager per gestire i dispositivi principali, vedere le funzionalità di Systems Manager nella Guida per l'AWS Systems Managerutente.

AWS Systems Manageroffre un livello di istanze standard e un livello di istanze avanzate per i nodi gestiti di Systems Manager. Se utilizzi Systems Manager per la prima volta, inizi dal livello delle istanze standard. Nel livello delle istanze standard, puoi registrare fino a 1.000 nodi gestiti per unità. Regione AWS Account AWS Se devi registrare più di 1.000 nodi gestiti in un unico account e regione o se devi utilizzare la funzionalità Session Manager, utilizza il livello delle istanze avanzate. Per ulteriori informazioni, consulta Configurazione dei livelli di istanza nella Guida per l'utente. AWS Systems Manager

Argomenti

Installare l'agente AWS Systems Manager

L'AWS Systems Manageragente (Systems Manager Agent) è un software Amazon che installi per consentire a Systems Manager di aggiornare, gestire e configurare i dispositivi core Greengrass, le istanze Amazon EC2 e altre risorse. L'agente elabora ed esegue le richieste dal servizio Systems Manager inCloud AWS. Quindi, l'agente invia le informazioni sullo stato e sul runtime al servizio Systems Manager. Per ulteriori informazioni, vedere Informazioni su Systems Manager Agent nella Guida AWS Systems Manager per l'utente.

AWSfornisce il Systems Manager Agent come componente Greengrass che è possibile distribuire sui dispositivi principali Greengrass per gestirli con Systems Manager. Il componente Systems Manager Agent installa il software Systems Manager Agent e registra il dispositivo principale come nodo gestito in Systems Manager. Segui i passaggi in questa pagina per completare i prerequisiti e distribuire il componente Systems Manager Agent su un dispositivo principale o un gruppo di dispositivi principali.

Argomenti

Fase 1: completare le fasi della configurazione generale di Systems Manager

Se non l'hai già fatto, completa i passaggi di configurazione generali per. AWS Systems Manager Per ulteriori informazioni, vedere Completare i passaggi generali di configurazione di Systems Manager nella Guida AWS Systems Manager per l'utente.

Fase 2: Creare un ruolo di servizio IAM per Systems Manager

L'agente Systems Manager utilizza un ruolo di servizio AWS Identity and Access Management (IAM) con cui comunicareAWS Systems Manager. Systems Manager assume questo ruolo per abilitare le funzionalità di Systems Manager su ogni dispositivo principale. Il componente Systems Manager Agent utilizza questo ruolo anche per registrare il dispositivo principale come nodo gestito di Systems Manager quando si distribuisce il componente. Se non l'hai già fatto, crea un ruolo di servizio Systems Manager da utilizzare per il componente Systems Manager Agent. Per ulteriori informazioni, consulta Creare un ruolo di servizio IAM per i dispositivi edge nella Guida per l'AWS Systems Managerutente.

Passaggio 3: Aggiungere le autorizzazioni al ruolo di scambio di token

I dispositivi core Greengrass utilizzano un ruolo di servizio IAM, chiamato ruolo di scambio di token, per interagire con AWS i servizi. Ogni dispositivo principale ha un ruolo di scambio di token che viene creato quando si installa il software AWS IoT Greengrass Core. Molti componenti Greengrass, come Systems Manager Agent, richiedono autorizzazioni aggiuntive per questo ruolo. Il componente agente Systems Manager richiede le seguenti autorizzazioni, che includono l'autorizzazione a utilizzare il ruolo in Fase 2: Creare un ruolo di servizio IAM per Systems Manager cui è stato creato.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iam::account-id:role/SSMServiceRole"
      ]
    },
    {
      "Action": [
        "ssm:AddTagsToResource",
        "ssm:RegisterManagedInstance"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
```

```
}
```

Se non l'hai già fatto, aggiungi queste autorizzazioni al ruolo di scambio di token del dispositivo principale per consentire il funzionamento di Systems Manager Agent. È possibile aggiungere una nuova politica al ruolo di scambio di token per concedere questa autorizzazione.

Per aggiungere autorizzazioni al ruolo di scambio di token (console)

1. Nel menu di navigazione della console IAM, scegli Ruoli.
2. Scegli il ruolo IAM che hai impostato come ruolo di scambio di token quando hai installato il software AWS IoT Greengrass Core. Se non hai specificato un nome per il ruolo di scambio di token quando hai installato il software AWS IoT Greengrass Core, è stato creato un ruolo denominato`GreengrassV2TokenExchangeRole`.
3. In Autorizzazioni, scegli Aggiungi autorizzazioni, quindi scegli Allega politiche.
4. Scegli Crea policy. La pagina Crea policy si apre in una nuova scheda del browser.
5. Nella pagina Create policy (Crea policy), eseguire le operazioni seguenti:

   a. Scegli JSON per aprire l'editor JSON.

   b. Incollare la seguente policy nell'editor JSON. Sostituisci *`SSM ServiceRole`* con il nome del ruolo di servizio in cui hai creato. Fase 2: Creare un ruolo di servizio IAM per Systems Manager

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iam::account-id:role/SSMServiceRole"
      ]
    },
    {
      "Action": [
        "ssm:AddTagsToResource",
        "ssm:RegisterManagedInstance"
      ],
      "Effect": "Allow",
```

```
      "Resource": "*"
    }
  ]
}
```

c. Scegliere Next: Tags (Successivo: Tag).

d. Scegliere Next:Review (Successivo: Rivedi).

e. Immettere un Name (Nome) per la policy, ad esempio **`GreengrassSSMAgentComponentPolicy`**.

f. Scegli Crea policy.

g. Passa alla scheda precedente del browser in cui è aperto il ruolo di scambio di token.

6. Nella pagina Aggiungi autorizzazioni, scegli il pulsante di aggiornamento, quindi seleziona la politica dell'agente Greengrass Systems Manager creata nel passaggio precedente.

7. Scegli Collega policy.

   I dispositivi principali che utilizzano questo ruolo di scambio di token ora sono autorizzati a interagire con il servizio Systems Manager.

Per aggiungere autorizzazioni al ruolo di scambio di token () AWS CLI

Per aggiungere una politica che conceda l'autorizzazione all'uso di Systems Manager

1. Crea un file chiamato `ssm-agent-component-policy.json` e copia il seguente codice JSON nel file. Sostituisci *`SSM ServiceRole`* con il nome del ruolo di servizio in cui hai creato. Fase 2: Creare un ruolo di servizio IAM per Systems Manager

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iam::account-id:role/SSMServiceRole"
      ]
    },
    {
```

```
      "Action": [
        "ssm:AddTagsToResource",
        "ssm:RegisterManagedInstance"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

2. Esegui il comando seguente per creare la politica dal documento di policy in`ssm-agent-component-policy.json`.

   Linux or Unix

   ```
   aws iam create-policy \
     --policy-name GreengrassSSMAgentComponentPolicy \
     --policy-document file://ssm-agent-component-policy.json
   ```

   Windows Command Prompt (CMD)

   ```
   aws iam create-policy ^
     --policy-name GreengrassSSMAgentComponentPolicy ^
     --policy-document file://ssm-agent-component-policy.json
   ```

   PowerShell

   ```
   aws iam create-policy `
     --policy-name GreengrassSSMAgentComponentPolicy `
     --policy-document file://ssm-agent-component-policy.json
   ```

   Copia la policy Amazon Resource Name (ARN) dai metadati della policy nell'output. Utilizzerai questo ARN per collegare questa policy al ruolo principale del dispositivo nel passaggio successivo.

3. Esegui il comando seguente per allegare la policy al ruolo di scambio di token.

   - Sostituisci *GreenGrassV2 TokenExchangeRole* con il nome del ruolo di scambio di token specificato al momento dell'installazione del AWS IoT Greengrass software Core. Se non hai

specificato un nome per il ruolo di scambio di token quando hai installato il software AWS IoT Greengrass Core, è stato creato un ruolo denominato. `GreengrassV2TokenExchangeRole`

- Sostituisci l'ARN della policy con l'ARN del passaggio precedente.

Linux or Unix

```
aws iam attach-role-policy \
  --role-name GreengrassV2TokenExchangeRole \
  --policy-arn
 arn:aws:iam::123456789012:policy/GreengrassSSMAgentComponentPolicy
```

Windows Command Prompt (CMD)

```
aws iam attach-role-policy ^
  --role-name GreengrassV2TokenExchangeRole ^
  --policy-arn
 arn:aws:iam::123456789012:policy/GreengrassSSMAgentComponentPolicy
```

PowerShell

```
aws iam attach-role-policy `
  --role-name GreengrassV2TokenExchangeRole `
  --policy-arn
 arn:aws:iam::123456789012:policy/GreengrassSSMAgentComponentPolicy
```

Se il comando non ha alcun risultato, è riuscito. I dispositivi principali che utilizzano questo ruolo di scambio di token ora sono autorizzati a interagire con il servizio Systems Manager.

Fase 4: Distribuire il componente Systems Manager Agent

Completare i seguenti passaggi per distribuire e configurare il componente Systems Manager Agent. È possibile distribuire il componente su un dispositivo single core o su un gruppo di dispositivi core.

Per distribuire il componente Systems Manager Agent (console)

1. Nel menu di navigazione AWS IoT Greengrassdella console, scegli Componenti.

2. Nella pagina Componenti, scegli la scheda Componenti pubblici, quindi scegli aws.greengrass.SystemsManagerAgent.
3. Nella pagina aws.greengrass.SystemsManagerAgent, scegli (Distribuisci).
4. Da Aggiungi alla distribuzione, scegli una distribuzione esistente da modificare oppure scegli di creare una nuova distribuzione, quindi scegli Avanti.
5. Se hai scelto di creare una nuova distribuzione, scegli il dispositivo principale o il gruppo di oggetti di destinazione per la distribuzione. Nella pagina Specificare la destinazione, in Obiettivo di distribuzione, scegli un dispositivo principale o un gruppo di oggetti, quindi scegli Avanti.
6. Nella pagina Seleziona componenti, verifica che il aws.greengrass.SystemsManagerAgentcomponente sia selezionato, scegli Avanti.
7. Nella pagina Configura componenti aws.greengrass.SystemsManagerAgent, selezionate e quindi effettuate le seguenti operazioni:

   a. Scegli Configura componente.

   b. Nella aws.greengrass.SystemsManagerAgent modalità Configura, in Aggiornamento della configurazione, in Configurazione da unire, inserisci il seguente aggiornamento di configurazione. Sostituisci *`SSM ServiceRole`* con il nome del ruolo di servizio in cui hai creato. Fase 2: Creare un ruolo di servizio IAM per Systems Manager

```
{
  "SSMRegistrationRole": "SSMServiceRole",
  "SSMOverrideExistingRegistration": false
}
```

> **Note**
>
> Se il dispositivo principale esegue già il Systems Manager Agent registrato con un'attivazione ibrida, passare `SSMOverrideExistingRegistration` a`true`. Questo parametro specifica se il componente Systems Manager Agent registra il dispositivo principale quando Systems Manager Agent è già in esecuzione sul dispositivo con un'attivazione ibrida.
> È inoltre possibile specificare tag (`SSMResourceTags`) da aggiungere al nodo gestito di Systems Manager creato dal componente Systems Manager Agent per il dispositivo principale. Per ulteriori informazioni, vedere Configurazione dei componenti Systems Manager Agent.

c. Scegli Conferma per chiudere la modalità, quindi scegli Avanti.

8. Nella pagina Configura impostazioni avanzate, mantieni le impostazioni di configurazione predefinite e scegli Avanti.
9. Nella pagina Review (Verifica), scegli Deploy (Distribuisci).

   Il completamento della distribuzione può richiedere fino a un minuto.

Per distribuire il componente Systems Manager Agent () AWS CLI

Per distribuire il componente Systems Manager Agent, create un documento di distribuzione che includa `aws.greengrass.SystemsManagerAgent` nell'`components`oggetto e specificate l'aggiornamento della configurazione per il componente. Segui le istruzioni riportate Creare distribuzione per creare una nuova distribuzione o modificare una distribuzione esistente.

Il seguente esempio di documento di distribuzione parziale specifica di utilizzare un ruolo di servizio denominato. `SSMServiceRole` Sostituisci *`SSM ServiceRole`* con il nome del ruolo di servizio in cui hai creato. Fase 2: Creare un ruolo di servizio IAM per Systems Manager

```
{
  ...,
  "components": {
    ...,
    "aws.greengrass.SystemsManagerAgent": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {
        "merge": "{\"SSMRegistrationRole\":\"SSMServiceRole\",
\"SSMOverrideExistingRegistration\":false}"
      }
    }
  }
}
```

> **Note**
>
> Se il dispositivo principale esegue già il Systems Manager Agent registrato con un'attivazione ibrida, passare `SSMOverrideExistingRegistration` `true`. Questo parametro specifica se il componente Systems Manager Agent registra il dispositivo principale quando Systems Manager Agent è già in esecuzione sul dispositivo con un'attivazione ibrida.

È inoltre possibile specificare tag (`SSMResourceTags`) da aggiungere al nodo gestito di Systems Manager creato dal componente Systems Manager Agent per il dispositivo principale. Per ulteriori informazioni, vedere Configurazione dei componenti Systems Manager Agent.

La distribuzione può richiedere alcuni minuti. È possibile utilizzare il AWS IoT Greengrass servizio per verificare lo stato della distribuzione e controllare i registri del software AWS IoT Greengrass Core e i registri dei componenti Systems Manager Agent per verificare che Systems Manager Agent funzioni correttamente. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Controllo dello stato di implementazione
- Monitora AWS IoT Greengrass i registri
- Visualizzazione dei log di Systems Manager Agent nella Guida per l'AWS Systems Managerutente

Se la distribuzione fallisce o Systems Manager Agent non viene eseguito, puoi risolvere i problemi di distribuzione su ogni dispositivo principale. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Risoluzione dei problemi AWS IoT Greengrass V2
- Risoluzione dei problemi di Systems Manager Agent nella guida AWS Systems Manager per l'utente

Fase 5: Verificare la registrazione del dispositivo principale con Systems Manager

Quando il componente Systems Manager Agent viene eseguito, registra il dispositivo principale come nodo gestito in Systems Manager. È possibile utilizzare la AWS IoT Greengrass console, la console Systems Manager e l'API Systems Manager per verificare che un dispositivo principale sia registrato come nodo gestito. I nodi gestiti sono anche chiamati istanze in alcune parti della AWS console e dell'API.

Per verificare la registrazione del dispositivo principale (AWS IoT Greengrassconsole)

1. Nel menu di navigazione della AWS IoT Greengrassconsole, scegli Dispositivi principali.
2. Scegli il dispositivo principale da verificare.

3. Nella pagina dei dettagli del dispositivo principale, trova la proprietà dell'AWS Systems Manageristanza. Se questa proprietà è presente e visualizza un collegamento alla console Systems Manager, il dispositivo principale viene registrato come nodo gestito.

   È inoltre possibile trovare la proprietà AWS Systems Managerping status per verificare lo stato del Systems Manager Agent sul dispositivo principale. Quando lo stato è Online, è possibile gestire il dispositivo principale con Systems Manager.

Per verificare la registrazione del dispositivo principale (console Systems Manager)

1. Nel menu di navigazione della console Systems Manager, scegli Fleet Manager.
2. In Nodi gestiti, procedi come segue:

   a. Aggiungi un filtro dove si trova il tipo di origine AWS::IoT::Thing.

   b. Aggiungi un filtro in cui Source ID è il nome del dispositivo principale da verificare.
3. Trova il dispositivo principale nella tabella Nodi gestiti. Se il dispositivo principale è nella tabella, è registrato come nodo gestito.

   È inoltre possibile trovare la proprietà ping status di Systems Manager Agent per verificare lo stato di Systems Manager Agent sul dispositivo principale. Quando lo stato è Online, è possibile gestire il dispositivo principale con Systems Manager.

Per verificare la registrazione del dispositivo principale (AWS CLI)

- Utilizzate l'DescribeInstanceInformationoperazione per ottenere l'elenco dei nodi gestiti che corrispondono a un filtro specificato. Esegui il comando seguente per verificare se un dispositivo principale è registrato come nodo gestito. Sostituiscilo *MyGreengrassCore*con il nome del dispositivo principale da verificare.

  ```
  aws ssm describe-instance-information --filter
   Key=SourceIds,Values=MyGreengrassCore Key=SourceTypes,Values=AWS::IoT::Thing
  ```

  La risposta contiene l'elenco dei nodi gestiti che corrispondono al filtro. Se l'elenco contiene un nodo gestito, il dispositivo principale viene registrato come nodo gestito. Puoi anche trovare altre informazioni sul nodo gestito del dispositivo principale nella risposta. Se la `PingStatus` proprietà è`Online`, è possibile gestire il dispositivo principale con Systems Manager.

Dopo aver verificato che un dispositivo principale sia registrato come nodo gestito in Systems Manager, è possibile utilizzare la console e l'API di Systems Manager per gestire quel dispositivo principale. Per ulteriori informazioni sulle funzionalità di Systems Manager che è possibile utilizzare per gestire i dispositivi core di Greengrass, vedere le funzionalità di Systems Manager nella Guida per l'AWS Systems Managerutente.

Disinstallazione dell'agente AWS Systems Manager

Se non desideri più gestire un dispositivo principale Greengrass conAWS Systems Manager, puoi annullare la registrazione del dispositivo principale da Systems Manager e disinstallare l'AWS Systems Manageragente (Systems Manager Agent) dal dispositivo.

Potrai registrare nuovamente un dispositivo principale in qualsiasi momento. A tale scopo, implementa nuovamente il componente Systems Manager Agent, che registra il dispositivo principale con Systems Manager al momento dell'installazione. Systems Manager gestisce la cronologia dei comandi per un dispositivo principale di cui è stata annullata la registrazione per 30 giorni.

Argomenti

Passaggio 1: annullare la registrazione del dispositivo principale da Systems Manager

È possibile utilizzare la console o l'API di Systems Manager per di cui è stata annullata la registrazione del dispositivo principale. Per ulteriori informazioni, vedere Annullamento della registrazione dei nodi gestiti nella Guida per l'AWS Systems Managerutente.

Passaggio 2: disinstallare il componente Systems Manager Agent

Dopo aver annullato la registrazione del dispositivo principale, disinstalla il componente Systems Manager Agent dal dispositivo. Per rimuovere un componente da un dispositivo principale di Greengrass, rivedi la distribuzione in cui è stato installato il componente e rimuovi il componente dalla distribuzione. Il softwareAWS IoT Greengrass Core disinstalla un componente quando nessuna delle distribuzioni di un dispositivo principale specifica quel componente. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.

Per disinstallare il componente Systems Manager Agent (console)

1. Nel menu di navigazione della AWS IoT Greengrassconsole, scegli Dispositivi Core.
2. Scegli il dispositivo principale in cui desideri disinstallare il componente Systems Manager Agent.
3. Nella pagina dei dettagli del dispositivo principale, scegli la scheda Distribuzioni.
4. Scegli la distribuzione che distribuisce il componente Systems Manager Agent sul dispositivo principale.
5. Nella pagina dei dettagli di distribuzione, scegli Revisione.
6. Nella modalità Revisione della distribuzione, scegli Revisione della distribuzione.
7. Nel passaggio 1: specifica l'obiettivo, scegli Avanti.
8. Nel passaggio 2: selezionare i componenti, cancellare la selezione del aws.greengrass.SystemsManagerAgentcomponente e quindi scegliere Avanti.
9. Nel passaggio 3: configura i componenti, scegli Avanti.
10. Nel passaggio 4: configura le impostazioni avanzate, scegli Avanti.
11. Nel passaggio 5: revisione, scegli Distribuisci.

Per disinstallare il componente Systems Manager Agent (CLI)

Per disinstallare il componente Systems Manager Agent, rivedi la distribuzione che lo distribuisce e rimuovilo dalla distribuzione. Per ulteriori informazioni, consulta Rivedi le distribuzioni.

Per il completamento della distribuzione possono essere necessari alcuni minuti. È possibile utilizzare ilAWS IoT Greengrass servizio per controllare lo stato della distribuzione. Per ulteriori informazioni, consulta Controllo dello stato di implementazione.

Passaggio 3: disinstallare il software Systems Manager Agent

Il software Systems Manager Agent continua a funzionare sul dispositivo principale dopo aver rimosso il componente Systems Manager Agent. Per rimuovere il software Systems Manager Agent, è possibile eseguire comandi sul dispositivo principale. Per ulteriori informazioni, vedere Disinstallare Systems Manager Agent dalle istanze Linux nella Guida per l'AWS Systems Managerutente.

Sicurezza in AWS IoT Greengrass

Per AWS, la sicurezza del cloud ha la massima priorità. In quanto cliente AWS, è possibile trarre vantaggio da un'architettura di data center e di rete progettata per soddisfare i requisiti delle organizzazioni più esigenti a livello di sicurezza.

La sicurezza è una responsabilità condivisa tra te e AWS. Il modello di responsabilità condivisa fa riferimento ad una sicurezza del cloud e nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che esegue i servizi AWS in Cloud AWS. AWS fornisce inoltre i servizi che è possibile utilizzare in modo sicuro. Revisori di terze parti testano regolarmente e verificano l'efficacia della nostra sicurezza nell'ambito dei Programmi di conformità AWS. Per informazioni sui programmi di conformità applicabili a AWS IoT Greengrass, consulta Servizi AWScoperti dal programma di conformità.
- Sicurezza nel cloud: la tua responsabilità è determinata dal servizio AWS che utilizzi. L'utente è anche responsabile di altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda e le leggi e le normative applicabili.

Quando utilizzato, AWS IoT Greengrass è anche responsabile della protezione dei dispositivi, della connessione di rete locale e delle chiavi private.

Questa documentazione serve a facilitare la comprensione dell'applicazione del modello di responsabilità condivisa quando si utilizza l'AWS IoT Greengrass. I seguenti argomenti illustrano come configurare l'AWS IoT Greengrass per soddisfare gli obiettivi di sicurezza e conformità. Scoprirai anche come utilizzare altri servizi di AWS per monitorare e proteggere le risorse AWS IoT Greengrass.

Argomenti

- Analisi della configurazione e delle vulnerabilità in AWS IoT Greengrass
- Integrità del codice inAWS IoT Greengrass V2
- AWS IoT Greengrass ed endpoint VPC dell'interfaccia (AWS PrivateLink)
- Best practice relative alla sicurezza di AWS IoT Greengrass

Protezione dei dati in AWS IoT Greengrass

Il modello di responsabilità condivisa di AWS si applica alla protezione dei dati in AWS IoT Greengrass. Come descritto in questo modello, AWSè responsabile della protezione dell'infrastruttura globale che esegue tutto l'Cloud AWS. L'utente è responsabile del controllo dei contenuti ospitati su questa infrastruttura. Inoltre, sei responsabile della configurazione della protezione e delle attività di gestione per i Servizi AWS che utilizzi. Per ulteriori informazioni sulla privacy dei dati, vedi le Domande frequenti sulla privacy dei dati. Per informazioni sulla protezione dei dati in Europa, consulta il post del blog relativo al Modello di responsabilità condivisa AWS e GDPR nel Blog sulla sicurezza AWS.

Per garantire la protezione dei dati, ti suggeriamo di proteggere le credenziali Account AWS e di configurare singoli utenti con AWS IAM Identity Center o AWS Identity and Access Management (IAM). In tal modo, a ogni utente verranno assegnate solo le autorizzazioni necessarie per svolgere i suoi compiti. Ti suggeriamo, inoltre, di proteggere i dati nei seguenti modi:

- Utilizza l'autenticazione a più fattori (MFA) con ogni account.
- Utilizza SSL/TLS per comunicare con le risorse AWS. È richiesto TLS 1.2 ed è consigliato TLS 1.3.
- Configura l'API e la registrazione delle attività degli utenti con AWS CloudTrail.
- Utilizza le soluzioni di crittografia AWS, insieme a tutti i controlli di sicurezza predefiniti in Servizi AWS.
- Utilizza i servizi di sicurezza gestiti avanzati, come Amazon Macie, che aiutano a individuare e proteggere i dati sensibili archiviati in Amazon S3.
- Se necessiti di moduli crittografici convalidati FIPS 140-2 quando accedi ad AWSattraverso un'interfaccia a riga di comando o un'API, utilizza un endpoint FIPS. Per ulteriori informazioni sugli endpoint FIPS disponibili, consulta il Federal Information Processing Standard (FIPS) 140-2.

Ti consigliamo vivamente di non inserire mai informazioni riservate o sensibili, ad esempio gli indirizzi e-mail dei clienti, nei tag o nei campi di testo in formato libero, ad esempio nel campo Nome. Questo vale quando si lavora con l'AWS IoT Greengrass e altri Servizi AWS utilizzando la console, l'API, la

AWS CLI o gli SDK di AWS. I dati inseriti nei tag o nei campi di testo in formato libero utilizzati per i nomi possono essere utilizzati per i la fatturazione o i log di diagnostica. Quando fornisci un URL a un server esterno, ti suggeriamo vivamente di non includere informazioni sulle credenziali nell'URL per convalidare la tua richiesta al server.

Per ulteriori informazioni sulla protezione delle informazioni sensibili inAWS IoT Greengrass, vederethe section called “Non registrare informazioni riservate”.

Per ulteriori informazioni sulla protezione dei dati, consulta il post del blog AWS Modello di responsabilità condivisa e GDPR su AWS Security Blog.

Argomenti

- Crittografia dei dati
- Integrazione della sicurezza hardware

Crittografia dei dati

AWS IoT Greengrassutilizza la crittografia per proteggere i dati durante il transito (su Internet o rete locale) e quando sono inattivi (archiviati inCloud AWS).

I dispositivi in un ambiente AWS IoT Greengrass spesso raccolgono dati inviati ai servizi AWS per ulteriori elaborazioni. Per ulteriori informazioni sulla crittografia dei dati su altri servizi AWS, consultare la documentazione di sicurezza per tale servizio.

Argomenti

- Crittografia dei dati in transito
- Crittografia dei dati a riposo
- Gestione delle chiavi per il dispositivo Core Greengrass

Crittografia dei dati in transito

AWS IoT Greengrassdispone di due modalità di comunicazione in cui i dati sono in transito:

- the section called “Dati in transito su Internet”. Comunicazione tra un nucleo di Greengrass eAWS IoT Greengrassvia Internet è criptato.
- the section called “Dati sul dispositivo core”. La comunicazione tra i componenti sul dispositivo core di Greengrass non è crittografata.

Dati in transito su Internet

AWS IoT Greengrass utilizza Transport Layer Security (TLS) per crittografare tutte le comunicazioni su Internet e sulla rete locale. Tutti i dati inviati adCloud AWSviene inviato tramite una connessione TLS utilizzando protocolli MQTT o HTTPS, pertanto è sicuro per impostazione predefinita.AWS IoT Greengrassutilizza ilAWS IoTmodello di sicurezza dei trasporti. Per ulteriori informazioni, consulta l'argomento relativo alla sicurezza del trasporto nella Guida per gli sviluppatori AWS IoT Core.

Dati sul dispositivo core

AWS IoT Greengrass non crittografa i dati scambiati localmente sul dispositivo core di Greengrass perché i dati non lasciano il dispositivo. Ciò include la comunicazione tra componenti definiti dall'utente,AWS IoTdispositivo SDK e componenti pubblici, come stream manager.

Crittografia dei dati a riposo

AWS IoT Greengrass memorizza i tuoi dati:

- the section called “Dati a riposo all'interno delCloud AWS”. Questi dati sono criptati.
- the section called “Dati inattivi sul core Greengrass”. Questi dati non sono crittografati (ad eccezione delle copie locali dei segreti).

Dati a riposo all'interno delCloud AWS

AWS IoT Greengrasscrittografa i dati dei clienti archiviati nelCloud AWS. Questi dati sono protetti utilizzando chiavi AWS KMS gestite da AWS IoT Greengrass.

Dati inattivi sul core Greengrass

AWS IoT Greengrass si basa sulle autorizzazioni dei file Unix e sulla crittografia completa del disco (se abilitata) per proteggere i dati inattivi sul core. È tua responsabilità proteggere il file system e il dispositivo.

Tuttavia, AWS IoT Greengrass crittografa le copie locali dei segreti recuperati da AWS Secrets Manager. Per ulteriori informazioni, consulta la .Secrets Managercomponente.

Gestione delle chiavi per il dispositivo Core Greengrass

È responsabilità del cliente garantire l'archiviazione sicura delle chiavi crittografiche (pubbliche e private) sul dispositivo core Greengrass.AWS IoT Greengrassutilizza chiavi pubbliche e private per il seguente scenario:

- La chiave client IoT viene utilizzata con il certificato IoT per autenticare l'handshake TLS (Transport Layer Security) quando un core Greengrass si connette a AWS IoT Core. Per ulteriori informazioni, consulta la pagina the section called “Autenticazione e autorizzazione del dispositivo” .

> **Note**
>
> La chiave e il certificato sono anche indicati come chiave privata principale e il certificato del dispositivo core.

Un dispositivo core Greengrass supporta l'archiviazione di chiavi private utilizzando le autorizzazioni del file system o unhardware security module. Se si utilizzano chiavi private basate su file system, si è responsabili della loro archiviazione sicura sul dispositivo core.

Integrazione della sicurezza hardware

> **Note**
>
> Questa funzionalità è disponibile per la versione 2.5.3 e successive del componente Greengrass nucleus. AWS IoT Greengrassattualmente non supporta questa funzionalità sui dispositivi Windows core.

È possibile configurare il software AWS IoT Greengrass Core per utilizzare un modulo di sicurezza hardware (HSM) tramite l'interfaccia PKCS #11. Questa funzionalità consente di archiviare in modo sicuro la chiave privata e il certificato del dispositivo in modo che non vengano esposti o duplicati nel software. È possibile archiviare la chiave privata e il certificato su un modulo hardware come un HSM o un Trusted Platform Module (TPM).

Il software AWS IoT Greengrass Core utilizza una chiave privata e un certificato X.509 per autenticare le connessioni ai servizi e. AWS IoT AWS IoT Greengrass Il componente secret manager utilizza questa chiave privata per crittografare e decrittografare in modo sicuro i segreti distribuiti su un dispositivo centrale Greengrass. Quando configurate un dispositivo principale per l'utilizzo di un HSM, questi componenti utilizzano la chiave privata e il certificato archiviati nell'HSM.

Il componente del broker Moquette MQTT memorizza anche una chiave privata per il certificato del server MQTT locale. Questo componente memorizza la chiave privata sul file system del

dispositivo nella cartella di lavoro del componente. Attualmente, AWS IoT Greengrass non supporta l'archiviazione di questa chiave privata o certificato in un HSM.

> **Tip**
>
> Cerca i dispositivi che supportano questa funzionalità nel AWSPartner Device Catalog.

Argomenti

Requisiti

È necessario soddisfare i seguenti requisiti per utilizzare un HSM su un dispositivo core Greengrass:

- Greengrass nucleus v2.5.3 o versione successiva installato sul dispositivo principale. È possibile scegliere una versione compatibile quando si installa il software AWS IoT Greengrass Core su un dispositivo principale.
- Il componente del provider PKCS #11 installato sul dispositivo principale. È possibile scaricare e installare questo componente quando si installa il software AWS IoT Greengrass Core su un dispositivo principale.
- Un modulo di sicurezza hardware che supporta lo schema di firma PKCS #1 v1.5 e le chiavi RSA con chiave RSA-2048 (o superiore) o chiavi ECC.

> **Note**
>
> Per utilizzare un modulo di sicurezza hardware con chiavi ECC, è necessario utilizzare Greengrass nucleus v2.5.6 o versione successiva.
> Per utilizzare un modulo di sicurezza hardware e un gestore segreto, è necessario utilizzare un modulo di sicurezza hardware con chiavi RSA.

- Una libreria di provider PKCS #11 che il software AWS IoT Greengrass Core può caricare in fase di esecuzione (usando libdl) per richiamare le funzioni PKCS #11. La libreria del provider PKCS #11 deve implementare le seguenti operazioni API PKCS #11:
  - `C_Initialize`
  - `C_Finalize`
  - `C_GetSlotList`
  - `C_GetSlotInfo`
  - `C_GetTokenInfo`
  - `C_OpenSession`
  - `C_GetSessionInfo`
  - `C_CloseSession`
  - `C_Login`
  - `C_Logout`
  - `C_GetAttributeValue`
  - `C_FindObjectsInit`
  - `C_FindObjects`
  - `C_FindObjectsFinal`
  - `C_DecryptInit`
  - `C_Decrypt`
  - `C_DecryptUpdate`
  - `C_DecryptFinal`
  - `C_SignInit`
  - `C_Sign`
  - `C_SignUpdate`
  - `C_SignFinal`
  - `C_GetMechanismList`
  - `C_GetMechanismInfo`
  - `C_GetInfo`
  - `C_GetFunctionList`
- Il modulo hardware deve essere risolvibile mediante l'etichetta dello slot, come definito nella specifica PKCS#11.

- È necessario archiviare la chiave privata e il certificato nell'HSM nello stesso slot e utilizzare la stessa etichetta e lo stesso ID dell'oggetto, se l'HSM supporta gli ID degli oggetti.
- Il certificato e la chiave privata devono essere risolvibili mediante etichette di oggetti.
- La chiave privata deve avere le seguenti autorizzazioni:
  - `sign`
  - `decrypt`
- (Facoltativo) Per utilizzare il componente Secret Manager, è necessario utilizzare la versione 2.1.0 o successiva e la chiave privata deve disporre delle seguenti autorizzazioni:
  - `unwrap`
  - `wrap`

Procedure ottimali per la sicurezza dell'hardware

Prendi in considerazione le seguenti best practice quando configuri la sicurezza hardware sui dispositivi core Greengrass.

- Generare chiavi private direttamente nell'HSM tramite il generatore di numeri casuali hardware interno. Questo approccio è più sicuro rispetto all'importazione di una chiave privata generata altrove, poiché la chiave privata rimane all'interno dell'HSM.
- Configura le chiavi private in modo che siano immutabili e proibisca l'esportazione.
- Utilizza lo strumento di provisioning consigliato dal fornitore di hardware HSM per generare una richiesta di firma del certificato (CSR) utilizzando la chiave privata protetta dall'hardware, quindi utilizza la console o l'API per generare un certificato client. AWS IoT

> **Note**
>
> La procedura consigliata in materia di sicurezza per ruotare le chiavi non si applica quando si generano chiavi private su un HSM.

Installa il software AWS IoT Greengrass Core con sicurezza hardware

Quando installi il software AWS IoT Greengrass Core, puoi configurarlo per utilizzare una chiave privata generata in un HSM. Questo approccio segue le migliori pratiche di sicurezza per generare la chiave privata nell'HSM, in modo che la chiave privata rimanga all'interno dell'HSM.

Per installare il software AWS IoT Greengrass Core con sicurezza hardware, procedi come segue:

1. Genera una chiave privata nell'HSM.
2. Crea una richiesta di firma del certificato (CSR) dalla chiave privata.
3. Crea un certificato dalla CSR. È possibile creare un certificato firmato da AWS IoT o da un'altra autorità di certificazione (CA) principale. Per ulteriori informazioni su come utilizzare un'altra CA root, consulta Create your own client certificate nella AWS IoT CoreDeveloper Guide.
4. Scarica il AWS IoT certificato e importalo nell'HSM.
5. Installa il software AWS IoT Greengrass Core da un file di configurazione che specifica di utilizzare il componente del provider PKCS #11 e la chiave privata e il certificato nell'HSM.

È possibile scegliere una delle seguenti opzioni di installazione per installare il software AWS IoT Greengrass Core con sicurezza hardware:

- Installazione manuale

  Scegli questa opzione per creare manualmente AWS le risorse richieste e configurare la sicurezza hardware. Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning manuale delle risorse.
- Installazione con provisioning personalizzato

  Scegliete questa opzione per sviluppare un'applicazione Java personalizzata che crei automaticamente AWS le risorse richieste e configuri la sicurezza hardware. Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning personalizzato delle risorse.

Attualmente, AWS IoT Greengrass non supporta l'installazione del software AWS IoT Greengrass Core con sicurezza hardware quando si installa con il provisioning automatico delle risorse o il provisioning del AWS IoT parco veicoli.

Configura la sicurezza hardware su un dispositivo principale esistente

È possibile importare la chiave privata e il certificato di un dispositivo principale in un HSM per configurare la sicurezza hardware.

Considerazioni

- È necessario disporre dell'accesso root al file system del dispositivo principale.
- In questa procedura, si spegne il software AWS IoT Greengrass Core, in modo che il dispositivo principale sia offline e non disponibile durante la configurazione della sicurezza hardware.

Per configurare la sicurezza hardware su un dispositivo principale esistente, procedi come segue:

1. Inizializza l'HSM.
2. Implementa il componente del provider PKCS #11 sul dispositivo principale.
3. Arresta il software AWS IoT Greengrass Core.
4. Importa la chiave privata e il certificato del dispositivo principale nell'HSM.
5. Aggiorna il file di configurazione del software AWS IoT Greengrass Core per utilizzare la chiave privata e il certificato nell'HSM.
6. Avvia il software AWS IoT Greengrass Core.

Fase 1: inizializzare il modulo di sicurezza hardware

Completa il passaggio seguente per inizializzare l'HSM sul tuo dispositivo principale.

Per inizializzare il modulo di sicurezza hardware

- Inizializza un token PKCS #11 nell'HSM e salva l'ID dello slot e il PIN utente relativi al token. Consulta la documentazione del tuo HSM per scoprire come inizializzare un token. L'ID dello slot e il PIN utente verranno utilizzati successivamente quando si distribuisce e si configura il componente del provider PKCS #11.

Fase 2: Implementazione del componente del provider PKCS #11

Completare i seguenti passaggi per distribuire e configurare il componente del provider PKCS #11. È possibile distribuire il componente su uno o più dispositivi principali.

Per distribuire il componente del provider PKCS #11 (console)

1. Nel menu di navigazione della AWS IoT Greengrassconsole, scegli Componenti.

2. Nella pagina Componenti, scegli la scheda Componenti pubblici, quindi scegli aws.greengrass.crypto.Pkcs11Provider.
3. Nella pagina aws.greengrass.crypto.Pkcs11Provider, scegli (Distribuisci).
4. Da Aggiungi alla distribuzione, scegli una distribuzione esistente da modificare oppure scegli di creare una nuova distribuzione, quindi scegli Avanti.
5. Se hai scelto di creare una nuova distribuzione, scegli il dispositivo principale o il gruppo di oggetti di destinazione per la distribuzione. Nella pagina Specificare la destinazione, in Obiettivo di distribuzione, scegli un dispositivo principale o un gruppo di oggetti, quindi scegli Avanti.
6. Nella pagina Seleziona componenti, in Componenti pubblici, seleziona aws.greengrass.crypto.Pkcs11Provider, quindi scegli Avanti.
7. Nella pagina Configura componenti aws.greengrass.crypto.Pkcs11Provider, selezionate e quindi effettuate le seguenti operazioni:

   a. Scegli Configura componente.

   b. Nella aws.greengrass.crypto.Pkcs11Provider modalità Configura, in Aggiornamento della configurazione, in Configurazione da unire, inserisci il seguente aggiornamento di configurazione. Aggiorna i seguenti parametri di configurazione con i valori per i dispositivi principali di destinazione. Specificate l'ID dello slot e il PIN utente in cui avete inizializzato in precedenza il token PKCS #11. La chiave privata e il certificato verranno importati in questo slot nell'HSM in un secondo momento.

      `name`

      Un nome per la configurazione PKCS #11.

      `library`

      Il percorso assoluto del file alla libreria dell'implementazione PKCS #11 che il software AWS IoT Greengrass Core può caricare con libdl.

      `slot`

      L'ID dello slot che contiene la chiave privata e il certificato del dispositivo. Questo valore è diverso dall'indice o dall'etichetta dello slot.

      `userPin`

      Il PIN utente da utilizzare per accedere allo slot.

```
{
  "name": "softhsm_pkcs11",
  "library": "/usr/lib/softhsm/libsofthsm2.so",
  "slot": 1,
  "userPin": "1234"
}
```

c. Scegli Conferma per chiudere la modalità, quindi scegli Avanti.

8. Nella pagina Configura impostazioni avanzate, mantieni le impostazioni di configurazione predefinite e scegli Avanti.
9. Nella pagina Review (Verifica), scegli Deploy (Distribuisci).

   Il completamento della distribuzione può richiedere fino a un minuto.

Per distribuire il componente provider PKCS #11 () AWS CLI

Per distribuire il componente del provider PKCS #11, create un documento di distribuzione che `aws.greengrass.crypto.Pkcs11Provider` includa l'`componentsoggetto` e specificate l'aggiornamento della configurazione per il componente. Segui le istruzioni riportate Creare distribuzione per creare una nuova distribuzione o modificare una distribuzione esistente.

Il seguente esempio di documento di distribuzione parziale specifica di distribuire e configurare il componente del provider PKCS #11. Aggiornate i seguenti parametri di configurazione con i valori per i dispositivi principali di destinazione. Salva l'ID dello slot e il PIN utente da utilizzare in seguito quando importi la chiave privata e il certificato nell'HSM.

`name`

Un nome per la configurazione PKCS #11.

`library`

Il percorso assoluto del file alla libreria dell'implementazione PKCS #11 che il software AWS IoT Greengrass Core può caricare con libdl.

`slot`

L'ID dello slot che contiene la chiave privata e il certificato del dispositivo. Questo valore è diverso dall'indice o dall'etichetta dello slot.

`userPin`

Il PIN utente da utilizzare per accedere allo slot.

```
{
  "name": "softhsm_pkcs11",
  "library": "/usr/lib/softhsm/libsofthsm2.so",
  "slot": 1,
  "userPin": "1234"
}
```

```
{
  ...,
  "components": {
    ...,
    "aws.greengrass.crypto.Pkcs11Provider": {
      "componentVersion": "2.0.0",
      "configurationUpdate": {
        "merge": "{\"name\":\"softhsm_pkcs11\",\"library\":\"/usr/lib/softhsm/
libsofthsm2.so\",\"slot\":1,\"userPin\":\"1234\"}"
      }
    }
  }
}
```

La distribuzione può richiedere alcuni minuti. È possibile utilizzare il AWS IoT Greengrass servizio per verificare lo stato della distribuzione. È possibile controllare i registri del software AWS IoT Greengrass Core per verificare che il componente del provider PKCS #11 venga distribuito correttamente. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Controllo dello stato di implementazione
- Monitora AWS IoT Greengrass i registri

Se la distribuzione non riesce, puoi risolvere i problemi relativi alla distribuzione su ogni dispositivo principale. Per ulteriori informazioni, consulta Risoluzione dei problemi AWS IoT Greengrass V2.

Passaggio 3: Aggiornare la configurazione sul dispositivo principale

Il software AWS IoT Greengrass Core utilizza un file di configurazione che specifica il funzionamento del dispositivo. Questo file di configurazione include dove trovare la chiave privata e il certificato

utilizzati dal dispositivo per connettersi a. Cloud AWS Completa i seguenti passaggi per importare la chiave privata e il certificato del dispositivo principale nell'HSM e aggiornare il file di configurazione per utilizzare l'HSM.

Per aggiornare la configurazione sul dispositivo principale per utilizzare la sicurezza hardware

1. Arresta il software AWS IoT Greengrass Core. Se hai configurato il software AWS IoT Greengrass Core come servizio di sistema con systemd, puoi eseguire il seguente comando per arrestare il software.

   ```
   sudo systemctl stop greengrass.service
   ```

2. Trova la chiave privata e i file di certificato del dispositivo principale.

   - Se hai installato il software AWS IoT Greengrass Core con provisioning automatico o fleet provisioning, la chiave privata esiste in*/greengrass/v2*/privKey.key, e il certificato esiste in. */greengrass/v2*/thingCert.crt
   - Se hai installato il software AWS IoT Greengrass Core con il provisioning manuale, la chiave privata esiste */greengrass/v2*/private.pem.key per impostazione predefinita e il certificato esiste per impostazione predefinita. */greengrass/v2*/device.pem.crt

   Puoi anche controllare le system.certificateFilePath proprietà system.privateKeyPath and */greengrass/v2*/config/effectiveConfig.yaml per trovare la posizione di questi file.

3. Importa la chiave privata e il certificato nell'HSM. Consulta la documentazione del tuo HSM per scoprire come importare chiavi private e certificati al suo interno. Importa la chiave privata e il certificato utilizzando l'ID dello slot e il PIN utente su cui hai inizializzato in precedenza il token PKCS #11. È necessario utilizzare la stessa etichetta dell'oggetto e lo stesso ID dell'oggetto per la chiave privata e il certificato. Salvate l'etichetta dell'oggetto che specificate quando importate ogni file. Questa etichetta viene utilizzata successivamente quando si aggiorna la configurazione del software AWS IoT Greengrass Core per utilizzare la chiave privata e il certificato nell'HSM.

4. Aggiorna la configurazione AWS IoT Greengrass Core per utilizzare la chiave privata e il certificato nell'HSM. Per aggiornare la configurazione, modificate il file di configurazione AWS IoT Greengrass Core ed eseguite il software AWS IoT Greengrass Core con il file di configurazione aggiornato per applicare la nuova configurazione.

   Esegui questa operazione:

a. Create un backup del file di configurazione AWS IoT Greengrass Core. Puoi utilizzare questo backup per ripristinare il dispositivo principale in caso di problemi durante la configurazione della sicurezza hardware.

```
sudo cp /greengrass/v2/config/effectiveConfig.yaml ~/ggc-config-backup.yaml
```

b. Apri il file di configurazione AWS IoT Greengrass Core in un editor di testo. Ad esempio, è possibile eseguire il comando seguente per utilizzare GNU nano per modificare il file. Sostituisci */greengrass/v2* con il percorso della cartella principale di Greengrass.

```
sudo nano /greengrass/v2/config/effectiveConfig.yaml
```

c. Sostituisci il valore di `system.privateKeyPath` con l'URI PKCS #11 per la chiave privata nell'HSM. Sostituisci *iotdevicekey* con l'etichetta dell'oggetto in cui hai importato in precedenza la chiave privata e il certificato.

```
pkcs11:object=iotdevicekey;type=private
```

d. Sostituisci il valore di `system.certificateFilePath` con l'URI PKCS #11 per il certificato nell'HSM. Sostituisci *iotdevicekey* con l'etichetta dell'oggetto in cui hai importato in precedenza la chiave privata e il certificato.

```
pkcs11:object=iotdevicekey;type=cert
```

Dopo aver completato questi passaggi, la `system` proprietà nel file di configurazione AWS IoT Greengrass Core dovrebbe essere simile all'esempio seguente.

```
system:
  certificateFilePath: "pkcs11:object=iotdevicekey;type=cert"
  privateKeyPath: "pkcs11:object=iotdevicekey;type=private"
  rootCaPath: "/greengrass/v2/rootCA.pem"
  rootpath: "/greengrass/v2"
  thingName: "MyGreengrassCore"
```

5. Applica la configurazione nel `effectiveConfig.yaml` file aggiornato. Esegui `Greengrass.jar` con il `--init-config` parametro in cui applicare la configurazione`effectiveConfig.yaml`. Sostituisci */greengrass/v2* con il percorso della cartella principale di Greengrass.

```
sudo java -Droot="/greengrass/v2" \
  -jar /greengrass/v2/alts/current/distro/lib/Greengrass.jar \
  --start false \
  --init-config /greengrass/v2/config/effectiveConfig.yaml
```

6. Avvia il software AWS IoT Greengrass Core. Se hai configurato il software AWS IoT Greengrass Core come servizio di sistema con systemd, puoi eseguire il seguente comando per avviare il software.

   ```
   sudo systemctl start greengrass.service
   ```

   Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core.

7. Controlla i registri del software AWS IoT Greengrass Core per verificare che il software si avvii e si connetta a. Cloud AWS Il software AWS IoT Greengrass Core utilizza la chiave privata e il certificato per connettersi ai AWS IoT Greengrass servizi AWS IoT and.

   ```
   sudo tail -f /greengrass/v2/logs/greengrass.log
   ```

   I seguenti messaggi di registro a livello di informazioni indicano che il software AWS IoT Greengrass Core si connette correttamente ai servizi AWS IoT andAWS IoT Greengrass.

   ```
   2021-12-06T22:47:53.702Z [INFO] (Thread-3)
    com.aws.greengrass.mqttclient.AwsIotMqttClient: Successfully connected to AWS IoT
    Core. {clientId=MyGreengrassCore5, sessionPresent=false}
   ```

8. (Facoltativo) Dopo aver verificato che il software AWS IoT Greengrass Core funzioni con la chiave privata e il certificato nell'HSM, eliminate la chiave privata e i file del certificato dal file system del dispositivo. Eseguite il comando seguente e sostituite i percorsi dei file con i percorsi della chiave privata e dei file dei certificati.

   ```
   sudo rm /greengrass/v2/privKey.key
   sudo rm /greengrass/v2/thingCert.crt
   ```

Utilizza hardware senza supporto PKCS #11

La libreria PKCS#11 viene in genere offerta dal fornitore hardware o è open source. Ad esempio, con l'hardware conforme allo standard (ad esempio TPM1.2), puoi utilizzare il software open source

esistente. Tuttavia, se il tuo hardware non dispone di un'implementazione della libreria PKCS #11 corrispondente o se desideri scrivere un provider PKCS #11 personalizzato, contatta il tuo rappresentante di Amazon Web Services Enterprise Support per domande relative all'integrazione.

Consulta anche

- Guida all'uso dell'interfaccia con token crittografici PKCS #11 versione 2.4.0
- RFC 7512
- PKCS #1: RSA Encryption versione 1.5

Autenticazione e autorizzazione del dispositivo per AWS IoT Greengrass

I dispositivi negli ambienti AWS IoT Greengrass utilizzano i certificati X.509 per l'autenticazione e le policy AWS IoT per l'autorizzazione. I certificati e le policy consentono ai dispositivi di connettersi in modo sicuro tra loro, AWS IoT Core e AWS IoT Greengrass.

I certificati X.509 sono certificati digitali che usano lo standard di infrastruttura a chiave pubblica X.509 per associare una chiave pubblica all'identità contenuta in un certificato. I certificati X.509 vengono rilasciati da un'entità attendibile denominata autorità di certificazione (CA). L'autorità di certificazione mantiene uno o più certificati speciali chiamati certificati CA, che usa per rilasciare certificati X.509. Solo l'autorità di certificazione ha accesso a certificati CA.

Le policy AWS IoT definiscono l'insieme di operazioni consentite per i dispositivi AWS IoT. In particolare, consentono e negano l'accesso alle operazioni del piano dei dati AWS IoT Core e AWS IoT Greengrass, ad esempio la pubblicazione di messaggi MQTT e il recupero delle copie shadow dei dispositivi.

Tutti i dispositivi richiedono una voce nel Registro di sistema AWS IoT Core e un certificato X.509 attivato con una policy AWS IoT allegata. I dispositivi rientrano in due categorie:

- Dispositivi core Greengrass

  I dispositivi core Greengrass utilizzano certificati e AWS IoT policy per connettersi a AWS IoT Core e. AWS IoT Greengrass I certificati e le politiche consentono inoltre di AWS IoT Greengrass distribuire componenti e configurazioni sui dispositivi principali.
- Dispositivi client

I dispositivi client MQTT utilizzano certificati e policy per la connessione AWS IoT Core e il AWS IoT Greengrass servizio. Ciò consente ai dispositivi client di utilizzare il AWS IoT Greengrass cloud discovery per trovare e connettersi a un dispositivo principale Greengrass. Un dispositivo client utilizza lo stesso certificato per connettersi al servizio AWS IoT Core cloud e ai dispositivi principali. I dispositivi client utilizzano anche le informazioni di rilevamento per l'autenticazione reciproca con il dispositivo principale. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

Certificati X.509

La comunicazione tra dispositivi principali e dispositivi client e tra dispositivi AWS IoT Core e/o AWS IoT Greengrass deve essere autenticata. Questa autenticazione si basa su certificati di dispositivo X.509 registrati e chiavi crittografiche.

In un ambiente AWS IoT Greengrass, i dispositivi utilizzano certificati con chiavi pubbliche e private per le seguenti connessioni TLS (Transport Layer Security):

- Il componente AWS IoT client del dispositivo principale Greengrass che si connette a AWS IoT Core e AWS IoT Greengrass tramite Internet.
- Dispositivi client che si connettono a AWS IoT Greengrass Internet per rilevare i dispositivi principali.
- Il componente broker MQTT sul core Greengrass si collega ai dispositivi Greengrass del gruppo tramite la rete locale.

AWS IoT Greengrassi dispositivi principali archiviano i certificati nella cartella principale di Greengrass.

Certificati dell'autorità di certificazione (CA)

I dispositivi core e i dispositivi client Greengrass scaricano un certificato CA root utilizzato per l'autenticazione con i servizi AWS IoT Core andAWS IoT Greengrass. Ti consigliamo di utilizzare un certificato di CA root di Amazon Trust Services (ATS), ad esempio Amazon Root CA 1. Per ulteriori informazioni, consulta Certificati emessi da una CA per l'autenticazione del server nella Guida per gli sviluppatori AWS IoT Core.

I dispositivi client scaricano anche un certificato CA del dispositivo principale Greengrass. Utilizzano questo certificato per convalidare il certificato del server MQTT sul dispositivo principale durante l'autenticazione reciproca.

Rotazione dei certificati sul broker MQTT locale

Quando si abilita il supporto per i dispositivi client, i dispositivi core Greengrass generano un certificato server MQTT locale che i dispositivi client utilizzano per l'autenticazione reciproca. Questo certificato è firmato dal certificato CA del dispositivo principale, che il dispositivo principale archivia nel AWS IoT Greengrass cloud. I dispositivi client recuperano il certificato CA del dispositivo principale quando scoprono il dispositivo principale. Utilizzano il certificato CA del dispositivo principale per verificare il certificato del server MQTT del dispositivo principale quando si connettono al dispositivo principale. Il certificato CA del dispositivo principale scade dopo 5 anni.

Per impostazione predefinita, il certificato del server MQTT scade ogni 7 giorni ed è possibile configurare questa durata tra 2 e 10 giorni. Questo periodo limitato si basa sulle best practice in materia di sicurezza. Questa rotazione aiuta a mitigare la minaccia che un utente malintenzionato rubi il certificato del server MQTT e la chiave privata per impersonare il dispositivo principale Greengrass.

Il dispositivo principale Greengrass ruota il certificato del server MQTT 24 ore prima della scadenza. Il dispositivo principale Greengrass genera un nuovo certificato e riavvia il broker MQTT locale. Quando ciò accade, tutti i dispositivi client collegati al dispositivo principale Greengrass vengono disconnessi. I dispositivi client possono riconnettersi al dispositivo principale Greengrass dopo un breve periodo di tempo.

Policy AWS IoT per operazioni del piano dei dati

Utilizza AWS IoT le policy per autorizzare l'accesso ai piani dati AWS IoT Core eAWS IoT Greengrass. Il piano AWS IoT Core dati fornisce operazioni per dispositivi, utenti e applicazioni. Queste operazioni includono la possibilità di connettersi AWS IoT Core e sottoscrivere argomenti. Il piano AWS IoT Greengrass dati fornisce le operazioni per i dispositivi Greengrass. Per ulteriori informazioni, consulta Operazioni di policy AWS IoT Greengrass V2. Queste operazioni includono la capacità di risolvere le dipendenze dei componenti e scaricare gli artefatti dei componenti pubblici.

Una AWS IoT policy è un documento JSON simile a una policy IAM. Contiene una o più istruzioni delle policy che specificano le proprietà seguenti:

- `Effect`. La modalità di accesso, che può essere `Allow` o`Deny`.
- `Action`. L'elenco delle azioni consentite o negate dalla politica.
- `Resource`. L'elenco delle risorse su cui l'azione è consentita o negata.

AWS IoTi criteri supportano i caratteri jolly * come caratteri jolly e trattano i caratteri jolly MQTT (+and#) come stringhe letterali. Per ulteriori informazioni sui caratteri jolly, vedete Uso dei * caratteri jolly negli ARN delle risorse nella Guida per l'utente. AWS Identity and Access Management

Per ulteriori informazioni, consulta Policy AWS IoT e Operazioni di policy AWS IoT nella Guida per gli sviluppatori AWS IoT Core.

> **Important**
>
> Le variabili Thing Policy (`iot:Connection.Thing.*`) non sono supportate nelle AWS IoT politiche per i dispositivi principali o nelle operazioni del piano dati Greengrass. Puoi invece utilizzare un carattere jolly che corrisponda a più dispositivi con nomi simili. Ad esempio, puoi specificare `MyGreengrassDevice*` di `MyGreengrassDevice1` abbinare e così via. `MyGreengrassDevice2`

> **Note**
>
> AWS IoT Coreconsente di allegare AWS IoT policy a gruppi di oggetti per definire le autorizzazioni per gruppi di dispositivi. Le policy relative ai gruppi di oggetti non consentono l'accesso alle operazioni del piano dati AWS IoT Greengrass. Per consentire a un oggetto di accedere a un'operazione del piano dati AWS IoT Greengrass, aggiungi l'autorizzazione a una policy AWS IoT collegata al certificato dell'oggetto.

Operazioni di policy AWS IoT Greengrass V2

AWS IoT Greengrass V2definisce le seguenti azioni politiche che i dispositivi core e i dispositivi client Greengrass possono utilizzare nelle AWS IoT politiche. Per specificare una risorsa per un'azione politica, utilizzi l'Amazon Resource Name (ARN) della risorsa.

Azioni principali del dispositivo

`greengrass:GetComponentVersionArtifact`

Concede l'autorizzazione a ottenere un URL predefinito per scaricare un elemento del componente pubblico o un elemento del componente Lambda.

Questa autorizzazione viene valutata quando un dispositivo principale riceve una distribuzione che specifica un componente pubblico o una Lambda con artefatti. Se il dispositivo principale contiene già l'artefatto, non lo scarica nuovamente.

Tipo di risorsa: `componentVersion`

Formato ARN delle risorse: `arn:aws:greengrass:`*`region`*`:`*`account-id`*`:components:`*`component-name`*`:versions:`*`component-version`*

`greengrass:ResolveComponentCandidates`

Concede l'autorizzazione a identificare un elenco di componenti che soddisfano i requisiti di componente, versione e piattaforma per una distribuzione. Se i requisiti sono in conflitto o non esistono componenti che soddisfino i requisiti, questa operazione restituisce un errore e la distribuzione non riesce sul dispositivo.

Questa autorizzazione viene valutata quando un dispositivo principale riceve una distribuzione che specifica i componenti.

Tipo di risorsa: Nessuno

Formato ARN delle risorse: *

`greengrass:GetDeploymentConfiguration`

Concede l'autorizzazione a ottenere un URL predefinito per scaricare un documento di distribuzione di grandi dimensioni.

Questa autorizzazione viene valutata quando un dispositivo principale riceve una distribuzione che specifica un documento di distribuzione più grande di 7 KB (se la distribuzione ha come obiettivo un oggetto) o 31 KB (se la distribuzione è destinata a un gruppo di oggetti). Il documento di distribuzione include le configurazioni dei componenti, le politiche di distribuzione e i metadati di distribuzione. Per ulteriori informazioni, consulta Implementazione AWS IoT Greengrass dei componenti sui dispositivi.

Questa funzionalità è disponibile per la versione 2.3.0 e successive del componente Greengrass nucleus.

Tipo di risorsa: Nessuno

Formato ARN delle risorse: *

`greengrass:ListThingGroupsForCoreDevice`

Concede l'autorizzazione a ottenere la gerarchia dei gruppi di oggetti di un dispositivo principale.

Questa autorizzazione viene verificata quando un dispositivo principale riceve una distribuzione da. AWS IoT Greengrass Il dispositivo principale utilizza questa azione per identificare se è stato rimosso da un gruppo di oggetti dopo l'ultima distribuzione. Se il dispositivo principale è stato rimosso da un gruppo di oggetti e quel gruppo di oggetti è l'obiettivo di una distribuzione sul dispositivo principale, il dispositivo principale rimuove i componenti installati da quella distribuzione.

Questa funzionalità è utilizzata dalla versione 2.5.0 e successive del componente Greengrass nucleus.

Tipo di risorsa: `thing` (dispositivo principale)

Formato ARN delle risorse: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`core-device-thing-name`*

`greengrass:VerifyClientDeviceIdentity`

Concede l'autorizzazione a verificare l'identità di un dispositivo client che si connette a un dispositivo principale.

Questa autorizzazione viene valutata quando un dispositivo principale esegue il componente di autenticazione del dispositivo client e riceve una connessione MQTT da un dispositivo client. Il dispositivo client presenta il certificato del dispositivo. AWS IoT Quindi, il dispositivo principale invia il certificato del dispositivo al servizio AWS IoT Greengrass cloud per verificare l'identità del dispositivo client. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

Tipo di risorsa: Nessuno

Formato ARN delle risorse: *

`greengrass:VerifyClientDeviceIoTCertificateAssociation`

Concede l'autorizzazione a verificare se un dispositivo client è associato a un AWS IoT certificato.

Questa autorizzazione viene valutata quando un dispositivo principale esegue il componente di autenticazione del dispositivo client e autorizza un dispositivo client a connettersi tramite MQTT. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

> **Note**
>
> Affinché un dispositivo principale possa utilizzare questa operazione, il ruolo di servizio Greengrass deve essere associato all'utente Account AWS e consentire l'`iot:DescribeCertificate`autorizzazione.

Tipo di risorsa: `thing` (dispositivo client)

Formato ARN delle risorse: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`client-device-thing-name`*

`greengrass:PutCertificateAuthorities`

Concede l'autorizzazione a caricare certificati di autorità di certificazione (CA) che i dispositivi client possono scaricare per verificare il dispositivo principale.

Questa autorizzazione viene valutata quando un dispositivo principale installa ed esegue il componente di autenticazione del dispositivo client. Questo componente crea un'autorità di certificazione locale e utilizza questa operazione per caricare i relativi certificati CA. I dispositivi client scaricano questi certificati CA quando utilizzano l'operazione Discover per trovare i dispositivi principali a cui connettersi. Quando i dispositivi client si connettono a un broker MQTT su un dispositivo principale, utilizzano questi certificati CA per verificare l'identità del dispositivo principale. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

Tipo di risorsa: Nessuno

Formato ARN: *

`greengrass:GetConnectivityInfo`

Concede l'autorizzazione a ottenere informazioni sulla connettività per un dispositivo principale. Queste informazioni descrivono come i dispositivi client possono connettersi al dispositivo principale.

Questa autorizzazione viene valutata quando un dispositivo principale installa ed esegue il componente di autenticazione del dispositivo client. Questo componente utilizza le informazioni di connettività per generare certificati CA validi da caricare sul servizio AWS IoT Greengrass cloud con l'operazione. PutCertificateAuthories I dispositivi client utilizzano questi certificati CA per verificare l'identità del dispositivo principale. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

È inoltre possibile utilizzare questa operazione sul piano di AWS IoT Greengrass controllo per visualizzare le informazioni di connettività per un dispositivo principale. Per ulteriori informazioni, consulta GetConnectivityInfo nella documentazione di riferimento dell'API AWS IoT Greengrass V1.

Tipo di risorsa: `thing` (dispositivo principale)

Formato ARN delle risorse: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`core-device-thing-name`*

`greengrass:UpdateConnectivityInfo`

Concede l'autorizzazione ad aggiornare le informazioni di connettività per un dispositivo principale. Queste informazioni descrivono come i dispositivi client possono connettersi al dispositivo principale.

Questa autorizzazione viene valutata quando un dispositivo principale esegue il componente del rilevatore IP. Questo componente identifica le informazioni necessarie ai dispositivi client per connettersi al dispositivo principale sulla rete locale. Quindi, questo componente utilizza questa operazione per caricare le informazioni di connettività sul servizio AWS IoT Greengrass cloud, in modo che i dispositivi client possano recuperare queste informazioni con l'operazione Discover. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

È inoltre possibile utilizzare questa operazione sul piano di AWS IoT Greengrass controllo per aggiornare manualmente le informazioni di connettività per un dispositivo principale. Per ulteriori informazioni, consulta UpdateConnectivityInfo nella documentazione di riferimento dell'API AWS IoT Greengrass V1.

Tipo di risorsa: `thing` (dispositivo principale)

Formato ARN delle risorse: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`core-device-thing-name`*

Azioni del dispositivo client

`greengrass:Discover`

Concede l'autorizzazione a scoprire le informazioni di connettività per i dispositivi principali a cui un dispositivo client può connettersi. Queste informazioni descrivono come il dispositivo client può connettersi ai dispositivi principali. Un dispositivo client può scoprire solo i dispositivi principali

a cui lo hai associato utilizzando l'BatchAssociateClientDeviceWithCoreDeviceoperazione. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.

Tipo di risorsa: `thing` (dispositivo client)

Formato ARN delle risorse: `arn:aws:iot:`*`region`*`:`*`account-id`*`:thing/`*`client-device-thing-name`*

Aggiorna la politica di un dispositivo principale AWS IoT

Puoi utilizzare le AWS IoT console AWS IoT Greengrass e o l'AWS IoTAPI per visualizzare e aggiornare la AWS IoT politica di un dispositivo principale.

> **Note**
>
> Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per il provisioning delle risorse, il tuo dispositivo principale dispone di una AWS IoT politica che consente l'accesso a tutte le AWS IoT Greengrass azioni ()`greengrass:*`. Puoi seguire questi passaggi per limitare l'accesso solo alle azioni utilizzate da un dispositivo principale.

Rivedi e aggiorna la AWS IoT politica di un dispositivo principale (console)

1. Nel menu di navigazione della AWS IoT Greengrassconsole, scegli Dispositivi principali.
2. Nella pagina Dispositivi principali, scegli il dispositivo principale da aggiornare.
3. Nella pagina dei dettagli del dispositivo principale, scegli il link all'oggetto del dispositivo principale. Questo link apre la pagina dei dettagli dell'oggetto nella AWS IoT console.
4. Nella pagina dei dettagli dell'oggetto, scegli Certificati.
5. Nella scheda Certificati, scegli il certificato attivo dell'oggetto.
6. Nella pagina dei dettagli del certificato, scegli Politiche.
7. Nella scheda Politiche, scegli la AWS IoT politica da rivedere e aggiornare. Puoi aggiungere le autorizzazioni richieste a qualsiasi policy allegata al certificato attivo del dispositivo principale.

   > **Note**
   >
   > Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per il provisioning delle risorse, hai due AWS IoT politiche. Ti consigliamo di scegliere

la politica denominata GreengrassV2IoTThingPolicy, se esiste. I dispositivi principali creati con il programma di installazione rapida utilizzano questo nome di policy per impostazione predefinita. Se aggiungi autorizzazioni a questa politica, le concedi anche ad altri dispositivi principali che utilizzano questa politica.

8. Nella panoramica della politica, scegli Modifica versione attiva.
9. Rivedi la politica e aggiungi, rimuovi o modifica le autorizzazioni secondo necessità.
10. Per impostare una nuova versione della politica come versione attiva, in Stato della versione della politica, seleziona Imposta la versione modificata come versione attiva per questa politica.
11. Scegli Salva come nuova versione.

Rivedi e aggiorna la AWS IoT politica di base del dispositivo (AWS CLI)

1. Elenca i principi fondamentali del AWS IoT dispositivo principale. I principali degli oggetti possono essere certificati di dispositivo X.509 o altri identificatori. Esegui il comando seguente e sostituiscilo *MyGreengrassCore*con il nome del dispositivo principale.

```
aws iot list-thing-principals --thing-name MyGreengrassCore
```

L'operazione restituisce una risposta che elenca i componenti principali del dispositivo principale.

```
{
    "principals": [
        "arn:aws:iot:us-west-2:123456789012:cert/certificateId"
    ]
}
```

2. Identifica il certificato attivo del dispositivo principale. Esegui il comando seguente e sostituisci *CertificateID* con l'ID di ogni certificato del passaggio precedente fino a trovare il certificato attivo. L'ID del certificato è la stringa esadecimale alla fine dell'ARN del certificato. L'`--query`argomento specifica di visualizzare solo lo stato del certificato.

```
aws iot describe-certificate --certificate-id certificateId --query
 'certificateDescription.status'
```

L'operazione restituisce lo stato del certificato sotto forma di stringa. Ad esempio, se il certificato è attivo, questa operazione genera un output`"ACTIVE"`.

3. Elenca le AWS IoT politiche allegate al certificato. Esegui il comando seguente e sostituisci l'ARN del certificato con l'ARN del certificato.

```
aws iot list-principal-policies --principal arn:aws:iot:us-
west-2:123456789012:cert/certificateId
```

L'operazione restituisce una risposta che elenca le AWS IoT politiche allegate al certificato.

```
{
    "policies": [
        {
            "policyName":
 "GreengrassTESCertificatePolicyMyGreengrassCoreTokenExchangeRoleAlias",
            "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassTESCertificatePolicyMyGreengrassCoreTokenExchangeRoleAlias"
        },
        {
            "policyName": "GreengrassV2IoTThingPolicy",
            "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy"
        }
    ]
}
```

4. Scegli la politica da visualizzare e aggiornare.

> **Note**
>
> Se hai utilizzato il programma di installazione del software AWS IoT Greengrass Core per il provisioning delle risorse, hai due AWS IoT politiche. Ti consigliamo di scegliere la politica denominata GreengrassV2IoTThingPolicy, se esiste. I dispositivi principali creati con il programma di installazione rapida utilizzano questo nome di policy per impostazione predefinita. Se aggiungi autorizzazioni a questa politica, le concedi anche ad altri dispositivi principali che utilizzano questa politica.

5. Scarica il documento della politica. Esegui il comando seguente e sostituisci *GreenGrassV2IoTThingPolicy* con il nome della politica.

```
aws iot get-policy --policy-name GreengrassV2IoTThingPolicy
```

L'operazione restituisce una risposta che contiene il documento della politica e altre informazioni sulla politica. Il documento di policy è un oggetto JSON serializzato come stringa.

```
{
    "policyName": "GreengrassV2IoTThingPolicy",
    "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
GreengrassV2IoTThingPolicy",
    "policyDocument": "{\
  \\"Version\\": \\"2012-10-17\\",\
  \\"Statement\\": [\
    {\
      \\"Effect\\": \\"Allow\\",\
      \\"Action\\": [\
                \\"iot:Connect\\",\
                \\"iot:Publish\\",\
                \\"iot:Subscribe\\",\
                \\"iot:Receive\\",\
                \\"greengrass:*\\"\
],\
      \\"Resource\\": \\"*\\"\
    }\
  ]\
}",
    "defaultVersionId": "1",
    "creationDate": "2021-02-05T16:03:14.098000-08:00",
    "lastModifiedDate": "2021-02-05T16:03:14.098000-08:00",
    "generationId":
 "f19144b798534f52c619d44f771a354f1b957dfa2b850625d9f1d0fde530e75f"
}
```

6. Utilizzate un convertitore online o un altro strumento per convertire la stringa del documento di policy in un oggetto JSON, quindi salvatela in un file denominato. `iot-policy.json`

   Ad esempio, se è installato lo strumento jq, è possibile eseguire il comando seguente per ottenere il documento di policy, convertirlo in un oggetto JSON e salvare il documento di policy come oggetto JSON.

   ```
   aws iot get-policy --policy-name GreengrassV2IoTThingPolicy --query
    'policyDocument' | jq fromjson >> iot-policy.json
   ```

7. Esamina il documento di policy e aggiungi, rimuovi o modifica le autorizzazioni secondo necessità.

   Ad esempio, su un sistema basato su Linux, è possibile eseguire il comando seguente per utilizzare GNU nano per aprire il file.

   ```
   nano iot-policy.json
   ```

   Al termine, il documento sulla policy potrebbe essere simile alla AWS IoTpolicy minima per i dispositivi principali.

8. Salva le modifiche come nuova versione della politica. Esegui il comando seguente e sostituisci *GreenGrassV2IoT ThingPolicy* con il nome della politica.

   ```
   aws iot create-policy-version --policy-name GreengrassV2IoTThingPolicy --policy-
   document file://iot-policy.json --set-as-default
   ```

   L'operazione restituisce una risposta simile all'esempio seguente se ha esito positivo.

   ```
   {
       "policyArn": "arn:aws:iot:us-west-2:123456789012:policy/
   GreengrassV2IoTThingPolicy",
       "policyDocument": "{\
     \\"Version\\": \\"2012-10-17\\",\
     \\"Statement\\": [\
       {\
         \\"Effect\\": \\"Allow\\",\
         \\"Action\\": [\
   \\t\\t\\"iot:Connect\\",\
   \\t\\t\\"iot:Publish\\",\
   \\t\\t\\"iot:Subscribe\\",\
   \\t\\t\\"iot:Receive\\",\
   \\t\\t\\"greengrass:*\\"\
         ],\
         \\"Resource\\": \\"*\\"\
       }\
     ]\
   }",
       "policyVersionId": "2",
       "isDefaultVersion": true
   }
   ```

AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali

Important

Le versioni successive del componente Greengrass nucleus richiedono autorizzazioni aggiuntive sulla policy minima. AWS IoT Potrebbe essere necessario aggiornare le AWS IoT politiche dei dispositivi principali per concedere autorizzazioni aggiuntive.

- I dispositivi principali che eseguono Greengrass nucleus v2.5.0 e versioni successive utilizzano l'`greengrass:ListThingGroupsForCoreDevice`autorizzazione per disinstallare i componenti quando si rimuove un dispositivo principale da un gruppo di oggetti.
- I dispositivi principali che eseguono Greengrass nucleus v2.3.0 e versioni successive utilizzano l'`greengrass:GetDeploymentConfiguration`autorizzazione per supportare documenti di configurazione di distribuzione di grandi dimensioni.

L'esempio di policy seguente include il set di operazioni minime necessarie per supportare le funzionalità di base di Greengrass per il dispositivo core.

- La `Connect` policy include il carattere `*` jolly dopo il nome dell'oggetto del dispositivo principale (ad esempio,). *`core-device-thing-name`** Il dispositivo principale utilizza lo stesso certificato del dispositivo per effettuare più sottoscrizioni simultaneeAWS IoT Core, ma l'ID client in una connessione potrebbe non corrispondere esattamente al nome dell'oggetto del dispositivo principale. Dopo i primi 50 abbonamenti, il dispositivo principale utilizza *`core-device-thing-name`*#*`number`* come ID client, dove *`number`* aumenta ogni 50 abbonamenti aggiuntivi. Ad esempio, quando un dispositivo principale denominato `MyCoreDevice` crea 150 abbonamenti simultanei, utilizza i seguenti ID client:
  - Abbonamenti da 1 a 50: `MyCoreDevice`
  - Abbonamenti da 51 a 100: `MyCoreDevice#2`
  - Abbonamenti da 101 a 150: `MyCoreDevice#3`

  La wildcard consente al dispositivo principale di connettersi quando utilizza questi ID client con un suffisso.
- La policy elenca gli argomenti MQTT e i filtri di argomento su cui il dispositivo core può pubblicare messaggi, sottoscrivere e ricevere messaggi, inclusi gli argomenti utilizzati per lo stato shadow. Per supportare lo scambio di messaggi tra AWS IoT Core i componenti Greengrass e i dispositivi client,

specifica gli argomenti e i filtri degli argomenti che desideri consentire. Per ulteriori informazioni, consulta Esempi di pubblicazione/sottoscrizione a policy nella Guida per gli sviluppatori AWS IoT Core.

- La politica concede l'autorizzazione alla pubblicazione dei dati di telemetria nel seguente argomento.

  ```
  $aws/things/core-device-thing-name/greengrass/health/json
  ```

  Puoi rimuovere questa autorizzazione per i dispositivi principali in cui disabiliti la telemetria. Per ulteriori informazioni, consulta Raccogli dati di telemetria sanitaria del sistema dai dispositivi principali AWS IoT Greengrass.

- La policy concede l'autorizzazione ad assumere un ruolo IAM tramite un alias di ruolo. AWS IoT Il dispositivo principale utilizza questo ruolo, chiamato ruolo di scambio di token, per acquisire AWS credenziali che può utilizzare per autenticare le richieste. AWS Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

  Quando installi il software AWS IoT Greengrass Core, crei e alleghi una seconda AWS IoT policy che include solo questa autorizzazione. Se includi questa autorizzazione nella AWS IoT politica principale del tuo dispositivo principale, puoi scollegare ed eliminare l'altra AWS IoT politica.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "arn:aws:iot:region:account-id:client/core-device-thing-name*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive",
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name/greengrass/health/json",
```

```
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name/greengrassv2/health/json",
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name/jobs/*",
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name/shadow/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-
thing-name/jobs/*",
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-
thing-name/shadow/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "iot:AssumeRoleWithCertificate",
            "Resource": "arn:aws:iot:region:account-id:rolealias/token-exchange-role-
alias-name"
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:GetComponentVersionArtifact",
                "greengrass:ResolveComponentCandidates",
                "greengrass:GetDeploymentConfiguration",
                "greengrass:ListThingGroupsForCoreDevice"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS IoTPolitica minima per supportare i dispositivi client

La seguente policy di esempio include il set minimo di azioni necessarie per supportare l'interazione con i dispositivi client su un dispositivo principale. Per supportare i dispositivi client, un dispositivo

principale deve disporre delle autorizzazioni previste in questa AWS IoT politica oltre alla AWS IoTpolitica minima per il funzionamento di base.

- La policy consente al dispositivo principale di aggiornare le proprie informazioni di connettività. Questa autorizzazione (`greengrass:UpdateConnectivityInfo`) è richiesta solo se si distribuisce il componente del rilevatore IP sul dispositivo principale.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name-gci/shadow/get"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-
thing-name-gci/shadow/update/delta",
                "arn:aws:iot:region:account-id:topicfilter/$aws/things/core-device-
thing-name-gci/shadow/get/accepted"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name-gci/shadow/update/delta",
                "arn:aws:iot:region:account-id:topic/$aws/things/core-device-thing-
name-gci/shadow/get/accepted"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:PutCertificateAuthorities",
                "greengrass:VerifyClientDeviceIdentity"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:VerifyClientDeviceIoTCertificateAssociation"
            ],
            "Resource": "arn:aws:iot:region:account-id:thing/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:GetConnectivityInfo",
                "greengrass:UpdateConnectivityInfo"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/core-device-thing-name"
            ]
        }
    ]
}
```

AWS IoTPolicy minima per i dispositivi client

La seguente politica di esempio include il set minimo di azioni necessarie affinché un dispositivo client rilevi i dispositivi principali su cui si connettono e comunicano tramite MQTT. La AWS IoT politica del dispositivo client deve includere l'`greengrass:Discover`azione per consentire al dispositivo di scoprire le informazioni di connettività per i dispositivi core Greengrass associati. Nella `Resource` sezione, specifica l'Amazon Resource Name (ARN) del dispositivo client, non l'ARN del dispositivo principale Greengrass.

- La policy consente la comunicazione su tutti gli argomenti MQTT. Per seguire le migliori pratiche di sicurezza, limitate le `iot:Publish` autorizzazioni e `iot:Receive` le autorizzazioni al set minimo di argomenti richiesti da un dispositivo client per il vostro caso d'uso. `iot:Subscribe`

- La policy consente al dispositivo di scoprire i dispositivi principali per qualsiasi AWS IoT cosa. Per seguire le migliori pratiche di sicurezza, limita l'`greengrass:Discover`autorizzazione all'AWS IoToggetto del dispositivo client o a un carattere jolly che corrisponda a un insieme di AWS IoT elementi.

  > **Important**
  >
  > Le variabili Thing Policy (`iot:Connection.Thing.*`) non sono supportate nelle AWS IoT politiche per i dispositivi principali o nelle operazioni del piano dati Greengrass. Puoi invece utilizzare un carattere jolly che corrisponda a più dispositivi con nomi simili. Ad esempio, puoi specificare `MyGreengrassDevice*` di `MyGreengrassDevice1` abbinare e così via. `MyGreengrassDevice2`

- La AWS IoT politica di un dispositivo client in genere non richiede autorizzazioni o `iot:DeleteThingShadow` azioni `iot:GetThingShadow``iot:UpdateThingShadow`, poiché il dispositivo principale Greengrass gestisce le operazioni di sincronizzazione degli shadow per i dispositivi client. Per consentire al dispositivo principale di gestire le ombre dei dispositivi client, verifica che la AWS IoT politica del dispositivo principale consenta queste azioni e che la `Resource` sezione includa gli ARN dei dispositivi client.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:Connect"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Publish"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "iot:Subscribe"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topicfilter/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iot:Receive"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:topic/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:Discover"
            ],
            "Resource": [
                "arn:aws:iot:region:account-id:thing/*"
            ]
        }
    ]
}
```

Gestione delle identità e degli accessi per l'AWS IoT Greengrass

AWS Identity and Access Management (IAM) è un Servizio AWS che consente agli amministratori di controllare in modo sicuro l'accesso alle risorse AWS. Gli amministratori IAM controllano chi è autenticato (accesso effettuato) e autorizzato (dispone di autorizzazioni) a utilizzare risorse AWS IoT Greengrass. IAM è un Servizio AWS che è possibile utilizzare senza alcun costo aggiuntivo.

> **Note**
>
> Questo argomento descrive i concetti e le funzionalità di IAM. Per informazioni sulle funzionalità IAM supportate daAWS IoT Greengrass, consultathe section called "Funzionamento di AWS IoT Greengrass con IAM".

Destinatari

Le modalità di utilizzo di AWS Identity and Access Management (IAM) cambiano in base alle operazioni eseguite in AWS IoT Greengrass.

Utente del servizio: se utilizzi il servizio AWS IoT Greengrass per eseguire il tuo lavoro, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. All'aumentare del numero di funzionalità AWS IoT Greengrass utilizzate per il lavoro, potrebbero essere necessarie ulteriori autorizzazioni. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità di AWS IoT Greengrass, consulta Risoluzione dei problemi di identità e accesso per AWS IoT Greengrass.

Amministratore del servizio: se sei il responsabile delle risorse AWS IoT Greengrass presso la tua azienda, probabilmente disponi dell'accesso completo a AWS IoT Greengrass. Il tuo compito è determinare le caratteristiche e le risorse AWS IoT Greengrass a cui gli utenti del servizio devono accedere. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per ulteriori informazioni su come la tua azienda può utilizzare IAM con AWS IoT Greengrass, consulta Funzionamento di AWS IoT Greengrass con IAM.

Amministratore IAM: un amministratore IAM potrebbe essere interessato a ottenere dei dettagli su come scrivere policy per gestire l'accesso a AWS IoT Greengrass. Per visualizzare policy basate su identità di AWS IoT Greengrass di esempio che puoi utilizzare in IAM, consulta Esempi di policy basate su identità per AWS IoT Greengrass.

Autenticazione con identità

L'autenticazione è la procedura di accesso ad AWS con le credenziali di identità. Devi essere autenticato (connesso a AWS) come utente root Utente root dell'account AWS, come utente IAM o assumere un ruolo IAM.

Puoi accedere ad AWS come identità federata utilizzando le credenziali fornite attraverso un'origine di identità. Gli utenti AWS IAM Identity Center (Centro identità IAM), l'autenticazione Single Sign-On (SSO) dell'azienda e le credenziali di Google o Facebook sono esempi di identità federate. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Se accedi ad AWS tramite la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, puoi accedere alla AWS Management Console o al portale di accesso AWS. Per ulteriori informazioni sull'accesso ad AWS, consulta la sezione Come accedere al tuo Account AWS nella Guida per l'utente di Accedi ad AWS.

Se accedi ad AWS in modo programmatico, AWS fornisce un Software Development Kit (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le richieste utilizzando le tue credenziali. Se non utilizzi gli strumenti AWS, devi firmare le richieste personalmente. Per ulteriori informazioni sulla firma delle richieste, consulta Firma delle richieste AWS nella Guida per l'utente IAM.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. AWS consiglia ad esempio di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza dell'account. Per ulteriori informazioni, consulta Autenticazione a più fattori nella Guida per l'utente di AWS IAM Identity Center e Utilizzo dell'autenticazione a più fattori (MFA) in AWS nella Guida per l'utente di IAM.

Utente root di un Account AWS

Quando crei un Account AWS, inizi con una singola identità di accesso che ha accesso completo a tutti i Servizi AWS e le risorse nell'account. Tale identità è detta utente root Account AWS ed è possibile accedervi con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzarle per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione Attività che richiedono le credenziali dell'utente root nella Guida per l'utente di IAM.

Utenti e gruppi IAM

Un utente IAM è una identità all'interno del tuo Account AWS che dispone di autorizzazioni specifiche per una singola persona o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, per casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina

Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine nella Guida per l'utente di IAM.

Un gruppo IAM è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato Amministratori IAM e concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta Quando creare un utente IAM (invece di un ruolo) nella Guida per l'utente di IAM.

Ruoli IAM

Un ruolo IAM è un'identità all'interno di un Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. È possibile assumere temporaneamente un ruolo IAM nella AWS Management Console mediante lo scambio di ruoli. È possibile assumere un ruolo chiamando un'azione AWS CLI o API AWS oppure utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta Utilizzo di ruoli IAM nella Guida per l'utente di IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- Accesso utente federato: per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta Creazione di un ruolo per un provider di identità di terza parte nella Guida per l'utente di IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per ulteriori informazioni sui set di autorizzazioni, consulta Set di autorizzazioni nella Guida per l'utente di AWS IAM Identity Center.
- Autorizzazioni utente IAM temporanee: un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- Accesso multi-account: è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento

principale per concedere l'accesso multi-account. Tuttavia, per alcuni dei Servizi AWS, è possibile collegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come proxy). Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta Differenza tra i ruoli IAM e le policy basate su risorse nella Guida per l'utente di IAM.

- Accesso multi-servizio: alcuni Servizi AWS utilizzano funzionalità in altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o archivi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
  - Inoltro delle sessioni di accesso (FAS): quando si utilizza un utente o un ruolo IAM per eseguire operazioni in AWS, tale utente o ruolo viene considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra azione in un servizio diverso. FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con il Servizio AWS richiedente, per effettuare richieste a servizi a valle. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che necessita di interazioni con altri Servizi AWS o risorse per essere portata a termine. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina Forward access sessions.
  - Ruolo di servizio: un ruolo di servizio è un ruolo IAM assunto da un servizio per eseguire operazioni per conto dell'utente. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS nella Guida per l'utente di IAM.
  - Ruolo collegato al servizio: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati ai servizi sono visualizzati nell'account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.
- Applicazioni in esecuzione su Amazon EC2: è possibile utilizzare un ruolo IAM per gestire credenziali temporanee per le applicazioni in esecuzione su un'istanza EC2 che eseguono richieste di AWS CLIo dell'API AWS. Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un ruolo AWS a un'istanza EC2, affinché sia disponibile per tutte le relative applicazioni, puoi creare un profilo dell'istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta Utilizzo di un ruolo IAM per concedere

autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2 nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta Quando creare un ruolo IAM (invece di un utente) nella Guida per l'utente di IAM.

Gestione dell'accesso con policy

Per controllare l'accesso a AWS è possibile creare policy e collegarle a identità o risorse AWS. Una policy è un oggetto in AWS che, quando associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste policy quando un principale IAM (utente, utente root o sessione ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle policy viene archiviata in AWSsotto forma di documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta Panoramica delle policy JSON nella Guida per l'utente di IAM.

Gli amministratori possono utilizzare le policy AWSJSON per specificare l'accesso ai diversi elementi. In altre parole, quale principale può eseguire azioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. Successivamente l'amministratore può aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'azione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dalla AWS Management Console, la AWS CLI o l'API AWS.

Policy basate su identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta Creazione di policy IAM nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono incorporate direttamente in un singolo utente, gruppo o ruolo. Le

policy gestite sono policy autonome che possono essere collegate a più utenti, gruppi e ruoli in Account AWS. Le policy gestite includono le policy gestite da AWS e le policy gestite dal cliente. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta Scelta fra policy gestite e policy inline nella Guida per l'utente di IAM.

Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile allegare a una risorsa. Gli esempi più comuni di policy basate su risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarle per controllare l'accesso a una risorsa specifica. Quando è allegata a una risorsa, una policy definisce le azioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario specificare un principale in una policy basata sulle risorse. I principali possono includere account, utenti, ruoli, utenti federati o Servizi AWS.

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy gestite da AWS da IAM in una policy basata su risorse.

Liste di controllo degli accessi (ACL)

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3, AWS WAF e Amazon VPC sono esempi di servizi che supportano le ACL. Per maggiori informazioni sulle ACL, consulta Panoramica delle liste di controllo degli accessi (ACL) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

Altri tipi di policy

AWS supporta altri tipi di policy meno comuni. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- Limiti delle autorizzazioni: un limite delle autorizzazioni è una funzione avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi

di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta Limiti delle autorizzazioni per le entità IAM nella Guida per l'utente di IAM.

- Policy di controllo dei servizi (SCP): le SCP sono policy JSON che specificano il numero massimo di autorizzazioni per un'organizzazione o unità organizzativa (OU) in AWS Organizations. AWS Organizationsè un servizio per il raggruppamento e la gestione centralizzata degli Account AWSmultipli di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. La SCP limita le autorizzazioni per le entità negli account membri, compreso ogni Utente root dell'account AWS. Per ulteriori informazioni su organizzazioni e policy SCP, consulta la pagina sulle Policy di controllo dei servizi nella Guida per l'utente di AWS Organizations.
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta Policy di sessione nella Guida per l'utente di IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per informazioni su come AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consulta Logica di valutazione delle policy nella Guida per l'utente di IAM.

Consulta anche

- the section called “Funzionamento di AWS IoT Greengrass con IAM”
- the section called “Esempi di policy basate su identità”
- the section called “Risoluzione dei problemi di identità e accesso”

Funzionamento di AWS IoT Greengrass con IAM

Prima di utilizzare IAM per gestire l'accesso aAWS IoT Greengrass, è necessario comprendere quali caratteristiche IAM sono disponibili per l'uso conAWS IoT Greengrass.

Funzionalità IAM	Supportata da Greengrass?
Policy basate sull'identità con autorizzazioni a livello di risorse	Sì
Policy basate su risorse	No
Liste di controllo accessi di rete (ACL)	No
Autorizzazione basata su tag	Sì
Credenziali temporanee	Sì
Ruoli collegati ai servizi	No
Ruoli di servizio	Sì

Per una panoramica generale del funzionamento di altriAWS servizi con IAM, consulta AWSServizi supportati da IAM nella Guida per l'utente di IAM.

Policy basate su identità per AWS IoT Greengrass

Con le policy basate su identità IAM puoi specificare operazioni e risorse consentite o rifiutate, nonché le condizioni in base alle quali le operazioni sono consentite o rifiutate. AWS IoT Greengrass supporta operazioni, risorse e chiavi di condizione specifiche. Per informazioni su tutti gli elementi utilizzati in una policy, consulta Documentazione di riferimento degli elementi delle policy JSON IAM nella Guida per l'utente di IAM.

Operazioni

Gli amministratori possono utilizzare le policy AWS JSON per specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le azioni che è possibile utilizzare per consentire o negare l'accesso a un criterio. Le operazioni della policy hanno spesso lo stesso nome dell'operazione API AWS. Ci sono alcune eccezioni, ad esempio le operazioni di sola autorizzazione che non hanno un'operazione API corrispondente. Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono chiamate operazioni dipendenti.

Includere le operazioni in una policy per concedere le autorizzazioni per eseguire l'operazione associata.

Le operazioni delle policy per AWS IoT Greengrass utilizzano il seguente prefisso `greengrass:` prima dell'operazione. Ad esempio, per consentire a qualcuno di utilizzare l'operazione`ListCoreDevices` API per elencarle dei dispositivi principaliAccount AWS, includere l'`greengrass:ListCoreDevices`operazione nella policy. Le istruzioni delle policy devono includere un elemento `Action` o `NotAction`. AWS IoT Greengrassdefinisce un proprio set di operazioni che descrivono le attività che puoi eseguire con questo servizio.

Per specificare più operazioni in una singola istruzione, elenarle tra parentesi (`[]`) e separarle con una virgola, come mostrato di seguito:

```
"Action": [
  "greengrass:action1",
  "greengrass:action2",
  "greengrass:action3"
]
```

È possibile utilizzare i caratteri jolly (`*`) per specificare più operazioni. Ad esempio, per specificare tutte le operazioni che iniziano con la parola `List`, includi la seguente operazione:

```
"Action": "greengrass:List*"
```

> **Note**
>
> Si consiglia di evitare l'uso di caratteri jolly per specificare tutte le operazioni disponibili per un servizio. Come procedura consigliata, è necessario concedere privilegi minimi e autorizzazioni di ambito ristretto in una policy. Per ulteriori informazioni, consulta the section called “Concedere autorizzazioni minime possibili”.

Per l'elenco completo delleAWS IoT Greengrass azioni, consulta Azioni definite daAWS IoT Greengrass nella Guida per l'utente IAM.

Risorse

Gli amministratori possono utilizzare le policy AWS JSON per specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Le istruzioni devono includere un elemento `Resource` o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il suo Amazon Resource Name (ARN). È possibile eseguire questa operazione per azioni che supportano un tipo di risorsa specifico, noto come autorizzazioni a livello di risorsa.

Per le operazioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

La tabella seguente contiene gli ARN della risorsa AWS IoT Greengrass che possono essere utilizzate nell'elemento `Resource` di un'istruzione delle policy. Per una mappatura delle autorizzazioni a livello di risorsa supportate perAWS IoT Greengrass le azioni, consulta Azioni definite daAWS IoT Greengrass nella Guida per l'utente IAM.

Alcune operazioni AWS IoT Greengrass (ad esempio, alcune operazioni di elenco), non possono essere eseguite su una risorsa specifica. In questi casi, è necessario utilizzare solo il carattere jolly.

```
"Resource": "*"
```

Per specificare più ARN di risorse in un'istruzione, elencali tra parentesi (`[]`) e separali con virgole, come segue:

```
"Resource": [
  "resource-arn1",
  "resource-arn2",
  "resource-arn3"
]
```

Per ulteriori informazioni sui formati ARN, consulta Amazon Resource Name (ARN) e spazi dei nomi del servizio AWS nella Riferimenti generali di Amazon Web Services.

Chiavi di condizione

Gli amministratori possono utilizzare le policy JSON AWS per specificare chi ha accesso a cosa. Cioè, quale principale può eseguire azioni su quali risorse, e in quali condizioni.

L'elemento `Condition` (o blocco `Condition`) consente di specificare le condizioni in cui un'istruzione è attiva. L'elemento `Condition` è facoltativo. Puoi compilare espressioni condizionali

che utilizzano operatori di condizione, ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta.

Se specifichi più elementi `Condition` in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione `AND` logica. Se specifichi più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'operazione `OR` logica. Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi autorizzare un utente IAM ad accedere a una risorsa solo se è stata taggata con il relativo nome utente IAM. Per ulteriori informazioni, consulta Elementi delle policy IAM: variabili e tag nella Guida per l'utente di IAM.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche per il servizio. Per visualizzare tutte le chiavi di condizione globali di AWS, consulta Chiavi di contesto delle condizioni globali di AWS nella Guida per l'utente di IAM.

Esempi

Per visualizzare esempi di policy basate su identità AWS IoT Greengrass, consulta the section called “Esempi di policy basate su identità”.

Policy basate su risorse per AWS IoT Greengrass

AWS IoT Greengrass non supporta le policy basate su risorse.

Liste di controllo accessi (ACL)

AWS IoT Greengrass non supporta ACL.

Autorizzazione basata su tag AWS IoT Greengrass

Puoi collegare i tag alle risorse AWS IoT Greengrass o passarli in una richiesta a AWS IoT Greengrass. Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'elemento condizione di una policy utilizzando le chiavi di condizione, `aws:ResourceTag/${TagKey}` o `aws:RequestTag/${TagKey}` `aws:TagKeys`. Per ulteriori informazioni, consulta Tagging delle risorse..

Ruoli IAM perAWS IoT Greengrass

Un ruolo IAM è un'entità all'interno di Account AWS che dispone di autorizzazioni specifiche.

Utilizzo di credenziali temporanee con AWS IoT Greengrass

Le credenziali temporanee vengono utilizzate per effettuare l'accesso utilizzando la federazione, assumere un ruolo IAM o assumere un ruolo multi-account. Per ottenere credenziali di sicurezza temporanee, eseguire una chiamata a operazioniAWS STS API quali AssumeRoleo GetFederationToken.

Sul core di Greengrass, le credenziali temporanee per il ruolo del dispositivo sono rese disponibili ai componenti Greengrass. Se i componenti utilizzano l'AWSSDK, non è necessario aggiungere logica per ottenere le credenziali perché l'AWSSDK lo fa per te.

Ruoli collegati ai servizi

AWS IoT Greengrass non supporta i ruoli collegati ai servizi.

Ruoli dei servizi

Questa caratteristica consente a un servizio di assumere un ruolo di servizio per conto dell'utente. Questo ruolo consente al servizio di accedere alle risorse in altri servizi per completare un'operazione per conto dell'utente. I ruoli dei servizi sono visualizzati nell'account IAM e sono di proprietà dell'account. Ciò significa che un amministratore IAM può modificare le autorizzazioni per questo ruolo. Tuttavia, questo potrebbe pregiudicare la funzionalità del servizio.

AWS IoT Greengrassi dispositivi principali utilizzano un ruolo di servizio per consentire ai componenti Greengrass e alle funzioni Lambda di accedere ad alcune delle tueAWS risorse per tuo conto. Per ulteriori informazioni, consulta the section called “Autorizza i dispositivi principali a interagire conAWSservizi”.

AWS IoT Greengrass utilizza un ruolo di servizio per accedere ad alcune risorse AWS per conto dell'utente. Per ulteriori informazioni, consulta Ruolo del servizio Greengrass.

Esempi di policy basate su identità per AWS IoT Greengrass

Per impostazione predefinita, gli utenti e i ruoli IAM non dispongono dell'autorizzazione per creare o modificare risorse AWS IoT Greengrass. Inoltre, non sono in grado di eseguire attività utilizzando la

AWS Management Console, AWS CLI o un'API AWS. Un amministratore IAM deve creare policy IAM che concedono a utenti e ruoli l'autorizzazione per eseguire operazioni API specifiche sulle risorse specificate di cui hanno bisogno. L'amministratore deve quindi allegare queste policy a utenti o IAM che richiedono tali autorizzazioni.

Best practice per le policy

Le policy basate su identità determinano se qualcuno può creare, accedere o eliminare risorse AWS IoT Greengrass nel tuo account. Queste operazioni possono comportare costi aggiuntivi per il proprio Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e suggerimenti:

- Nozioni di base sulle policy gestite da AWS e passaggio alle autorizzazioni con privilegio minimo: per le informazioni di base su come concedere autorizzazioni a utenti e carichi di lavoro, utilizza le policy gestite da AWS che concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo Account AWS. Ti consigliamo pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente di AWS specifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta Policy gestite da AWS o Policy gestite da AWS per le funzioni di processo nella Guida per l'utente di IAM.
- Applica le autorizzazioni con privilegio minimo: quando imposti le autorizzazioni con le policy IAM, concedi solo le autorizzazioni richieste per eseguire un'attività. Puoi farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegi minimi. Per ulteriori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta Policy e autorizzazioni in IAM nella Guida per l'utente di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso: per limitare l'accesso a operazioni e risorse puoi aggiungere una condizione alle tue policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. Puoi inoltre utilizzare le condizioni per concedere l'accesso alle operazioni di servizio, ma solo se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta la sezione Elementi delle policy JSON di IAM: condizione nella Guida per l'utente IAM.
- Utilizzo di IAM Access Analyzer per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano al linguaggio della policy IAM (JSON) e alle best practice di IAM. IAM Access Analyzer fornisce oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per ulteriori informazioni, consulta Convalida delle policy per IAM Access Analyzer nella Guida per l'utente di IAM.

- Richiesta dell'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o utenti root nel tuo Account AWS, attiva MFA per una maggiore sicurezza. Per richiedere l'AMF quando vengono chiamate le operazioni API, aggiungi le condizioni MFA alle policy. Per ulteriori informazioni, consulta Configurazione dell'accesso alle API protetto con MFA nella Guida per l'utente di IAM.

Per maggiori informazioni sulle best practice in IAM, consulta Best practice di sicurezza in IAM nella Guida per l'utente di IAM.

Esempi di policy

Nell'esempio riportato di seguito le policy definite dal cliente concedono autorizzazioni per scenari comuni.

Esempi

- Consentire agli utenti di visualizzare le loro autorizzazioni

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consultare Creazione di policy nella scheda JSON nella Guida per l'utente di IAM.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono allegate alla relativa identità utente. La policy include le autorizzazioni per completare questa operazione sulla console o a livello di programmazione utilizzando la AWS CLI o l'API AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
```

```
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Autorizza i dispositivi principali a interagire conAWSservizi

AWS IoT Greengrassi dispositivi principali utilizzano il kitAWS IoT Coreprovider di credenziali per autorizzare le chiamate aAWSServizi . LaAWS IoT Coreprovider di credenziali consente ai dispositivi di utilizzare i certificati X.509 come identità univoca del dispositivo da autenticareAWSrichieste. In questo modo, non è più necessario memorizzare unaAWSID chiave di accesso e chiave di accesso segreta sul tuoAWS IoT Greengrassdispositivi core. Per ulteriori informazioni, consultaAutorizzazione di chiamate dirette aAWSserviziinellaAWS IoT CoreGuida per lo Sviluppatore.

Quando esegui ilAWS IoT GreengrassSoftware di base, è possibile scegliere di eseguire il provisioningAWSrisorse richieste dal dispositivo principale. Ciò include il kitAWS Identity and Access Managementruolo (IAM) che il tuo dispositivo principale assume tramiteAWS IoT Corefornitore di credenziali. Utilizzo dell'`--provision true`argomento per configurare un ruolo e criteri che consentono al dispositivo principale di diventare temporaneiAWSCredenziali . Questo argomento configura anche unAWS IoTAlias del ruolo che punta a questo ruolo IAM. È possibile specificare il nome del ruolo IAM eAWS IoTAlias del ruolo da utilizzare. Se si specifica`--provision true`senza questi altri parametri di nome, il dispositivo principale Greengrass crea e utilizza le seguenti risorse predefinite:

- Ruolo IAM:`GreengrassV2TokenExchangeRole`

Questo ruolo ha un criterio denominato`GreengrassV2TokenExchangeRoleAccesse` una relazione di trust che consente`credentials.iot.amazonaws.com`per assumere il ruolo. Il criterio include le autorizzazioni minime per il dispositivo principale.

> **Important**
>
> Questo criterio non include l'accesso ai file nei bucket S3. È necessario aggiungere le autorizzazioni al ruolo per consentire ai dispositivi principali di recuperare gli artifact dei componenti dai bucket S3. Per ulteriori informazioni, consulta la pagina Consenti l'accesso ai bucket S3 per gli artefatti dei componenti .

- AWS IoTAlias del ruolo:`GreengrassV2TokenExchangeRoleAlias`

  Questo alias di ruolo si riferisce al ruolo IAM.

Per ulteriori informazioni, consulta la pagina Fase 3: installare il softwareAWS IoT Greengrass Core .

È inoltre possibile impostare l'alias di ruolo per un dispositivo core esistente. A questo proposito, configurare il kit`iotRoleAlias`parametro di configurazione delComponente nucleo Greengrass.

È possibile acquisire temporaneamenteAWScredenziali per eseguire questo ruolo IAMAWSoperazioni nei tuoi componenti personalizzati. Per ulteriori informazioni, consulta la pagina Interagisci con AWS i servizi .

Argomenti

- Autorizzazioni del ruolo del servizio per i dispositivi principali
- Consenti l'accesso ai bucket S3 per gli artefatti dei componenti

Autorizzazioni del ruolo del servizio per i dispositivi principali

Il ruolo consente al seguente servizio di assumere il ruolo:

- `credentials.iot.amazonaws.com`

Se utilizzi il pluginAWS IoT GreengrassSoftware di base per creare questo ruolo, utilizza i seguenti criteri di autorizzazione per consentire ai dispositivi principali di connettersi e inviare i log aAWS. Il nome del criterio per impostazione predefinita è il nome del ruolo IAM che termina

conAccess. Ad esempio, se si utilizza il nome del ruolo IAM predefinito, il nome di questo criterio èGreengrassV2TokenExchangeRoleAccess.

Greengrass nucleus v2.5.0 and later

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

v2.4.x

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:DescribeCertificate",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

Earlier than v2.4.0

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:DescribeCertificate",
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogStreams",
        "iot:Connect",
        "iot:Publish",
        "iot:Subscribe",
        "iot:Receive",
        "s3:GetBucketLocation"
      ],
      "Resource": "*"
    }
  ]
}
```

Consenti l'accesso ai bucket S3 per gli artefatti dei componenti

Il ruolo del dispositivo principale predefinito non consente ai dispositivi core di accedere ai bucket S3. Per distribuire componenti che dispongono di artifact nei bucket S3, è necessario aggiungere`s3:GetObject`autorizzazione per consentire ai dispositivi principali di scaricare gli artefatti dei componenti. È possibile aggiungere un nuovo criterio al ruolo principale del dispositivo per concedere questa autorizzazione.

Per aggiungere una policy che consenta l'accesso agli artefatti dei componenti in Amazon S3

1. Crea un file denominato`component-artifact-policy.json`e copia il seguente JSON nel file. Questa policy consente l'accesso a tutti i file in un bucket S3. Replace (Sostituisci)*`SECCHIELLO DOC-ESEMPIO-SECCHIO`*con il nome del bucket S3 bucket per consentire l'accesso al dispositivo principale.

   ```
   {
     "Version": "2012-10-17",
   ```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::DOC-EXAMPLE-BUCKET/*"
    }
  ]
}
```

2. Esegui il comando seguente per creare la policy dal documento della policy in`component-artifact-policy.json`.

   Linux or Unix

   ```
   aws iam create-policy \
     --policy-name MyGreengrassV2ComponentArtifactPolicy \
     --policy-document file://component-artifact-policy.json
   ```

   Windows Command Prompt (CMD)

   ```
   aws iam create-policy ^
     --policy-name MyGreengrassV2ComponentArtifactPolicy ^
     --policy-document file://component-artifact-policy.json
   ```

   PowerShell

   ```
   aws iam create-policy `
     --policy-name MyGreengrassV2ComponentArtifactPolicy `
     --policy-document file://component-artifact-policy.json
   ```

   Copiare la policy Amazon Resource Name (ARN) della policy dai metadati della policy nell'output. Usare l'ARN per collegare questa policy al ruolo del dispositivo principale nella fase successiva.

3. Esegui il comando seguente per collegare la policy al ruolo del dispositivo principale. Replace (Sostituisci)*Green Grass V2 Token Ruolo di cambio*con il nome del ruolo che hai specificato quando hai eseguito laAWS IoT GreengrassSoftware core. Sostituisci l'ARN della policy con l'ARN della fase precedente.

Linux or Unix

```
aws iam attach-role-policy \
  --role-name GreengrassV2TokenExchangeRole \
  --policy-arn
 arn:aws:iam::123456789012:policy/MyGreengrassV2ComponentArtifactPolicy
```

Windows Command Prompt (CMD)

```
aws iam attach-role-policy ^
  --role-name GreengrassV2TokenExchangeRole ^
  --policy-arn
 arn:aws:iam::123456789012:policy/MyGreengrassV2ComponentArtifactPolicy
```

PowerShell

```
aws iam attach-role-policy `
  --role-name GreengrassV2TokenExchangeRole `
  --policy-arn
 arn:aws:iam::123456789012:policy/MyGreengrassV2ComponentArtifactPolicy
```

Se il comando non ha output, è riuscito e il dispositivo principale può accedere agli artefatti caricati in questo bucket S3.

Policy IAM minima per l'installatore per il provisioning delle risorse

Quando installi il software AWS IoT Greengrass Core, puoi fornire AWS le risorse necessarie, come un AWS IoT oggetto e un ruolo IAM per il tuo dispositivo. Puoi anche implementare strumenti di sviluppo locale sul dispositivo. Il programma di installazione richiede AWS credenziali per poter eseguire queste azioni nel tuo. Account AWS Per ulteriori informazioni, consulta Installare il software AWS IoT Greengrass Core..

La seguente politica di esempio include il set minimo di azioni richieste dal programma di installazione per fornire queste risorse. Queste autorizzazioni sono necessarie se si specifica l'`--provision`argomento per l'installatore. Sostituisci *`account-id`* con il tuo Account AWS ID e sostituisci *`greengrassV2_TokenExchangeRole`* con il nome del ruolo di scambio di token specificato con l'argomento installer. `--tes-role-name`

> **Note**
>
> L'informativa `DeployDevTools` sulla politica è richiesta solo se si specifica l'argomento per l'installatore. `--deploy-dev-tools`

Greengrass nucleus v2.5.0 and later

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateTokenExchangeRole",
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreatePolicy",
                "iam:CreateRole",
                "iam:GetPolicy",
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::account-id:role/GreengrassV2TokenExchangeRole",
                "arn:aws:iam::account-
id:policy/GreengrassV2TokenExchangeRoleAccess",
                "arn:aws:iam::aws:policy/GreengrassV2TokenExchangeRoleAccess"
            ]
        },
        {
            "Sid": "CreateIoTResources",
            "Effect": "Allow",
            "Action": [
                "iot:AddThingToThingGroup",
                "iot:AttachPolicy",
                "iot:AttachThingPrincipal",
                "iot:CreateKeysAndCertificate",
                "iot:CreatePolicy",
                "iot:CreateRoleAlias",
                "iot:CreateThing",
                "iot:CreateThingGroup",
                "iot:DescribeEndpoint",
                "iot:DescribeRoleAlias",
```

```
                "iot:DescribeThingGroup",
                "iot:GetPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DeployDevTools",
            "Effect": "Allow",
            "Action": [
                "greengrass:CreateDeployment",
                "iot:CancelJob",
                "iot:CreateJob",
                "iot:DeleteThingShadow",
                "iot:DescribeJob",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:GetThingShadow",
                "iot:UpdateJob",
                "iot:UpdateThingShadow"
            ],
            "Resource": "*"
        }
    ]
}
```

Earlier than v2.5.0

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CreateTokenExchangeRole",
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreatePolicy",
                "iam:CreateRole",
                "iam:GetPolicy",
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::account-id:role/GreengrassV2TokenExchangeRole",
```

```
                "arn:aws:iam::account-
id:policy/GreengrassV2TokenExchangeRoleAccess",
                "arn:aws:iam::aws:policy/GreengrassV2TokenExchangeRoleAccess"
            ]
        },
        {
            "Sid": "CreateIoTResources",
            "Effect": "Allow",
            "Action": [
                "iot:AddThingToThingGroup",
                "iot:AttachPolicy",
                "iot:AttachThingPrincipal",
                "iot:CreateKeysAndCertificate",
                "iot:CreatePolicy",
                "iot:CreateRoleAlias",
                "iot:CreateThing",
                "iot:CreateThingGroup",
                "iot:DescribeEndpoint",
                "iot:DescribeRoleAlias",
                "iot:DescribeThingGroup",
                "iot:GetPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DeployDevTools",
            "Effect": "Allow",
            "Action": [
                "greengrass:CreateDeployment",
                "iot:CancelJob",
                "iot:CreateJob",
                "iot:DeleteThingShadow",
                "iot:DescribeJob",
                "iot:DescribeThing",
                "iot:DescribeThingGroup",
                "iot:GetThingShadow",
                "iot:UpdateJob",
                "iot:UpdateThingShadow"
            ],
            "Resource": "*"
        }
    ]
}
```

Ruolo del servizio Greengrass

Il ruolo di servizio Greengrass è un ruolo di servizio AWS Identity and Access Management (IAM) che AWS IoT Greengrass autorizza l'accesso alle risorse AWS dei servizi per conto dell'utente. Questo ruolo consente di AWS IoT Greengrass verificare l'identità dei dispositivi client e gestire le informazioni principali sulla connettività dei dispositivi.

> **Note**
>
> AWS IoT Greengrass V1utilizza questo ruolo anche per eseguire attività essenziali. Per ulteriori informazioni, consulta il ruolo del servizio Greengrass nella AWS IoT Greengrass V1Developer Guide.

Per consentire l'accesso AWS IoT Greengrass alle tue risorse, il ruolo di servizio Greengrass deve essere associato a te Account AWS e specificato AWS IoT Greengrass come entità affidabile. Il ruolo deve includere la politica AWSGreengrassResourceAccessRolePolicygestita o una politica personalizzata che definisca autorizzazioni equivalenti per le AWS IoT Greengrass funzionalità utilizzate. AWSmantiene questa politica, che definisce l'insieme di autorizzazioni AWS IoT Greengrass utilizzate per accedere AWS alle risorse. Per ulteriori informazioni, consulta AWSPolicy gestita: AWSGreengrassResourceAccessRolePolicy.

Puoi riutilizzare lo stesso ruolo del servizio Greengrass Regioni AWS in tutto il mondo, ma devi associarlo al tuo account Regione AWS ovunque lo utilizzi. AWS IoT Greengrass Se il ruolo di servizio non è configurato nell'attuale versioneRegione AWS, i dispositivi principali non riescono a verificare i dispositivi client e ad aggiornare le informazioni di connettività.

Le sezioni seguenti descrivono come creare e gestire il ruolo di servizio Greengrass con o. AWS Management Console AWS CLI

Argomenti

- Gestire il ruolo del servizio Greengrass (console)
- Gestione del ruolo di servizio Greengrass (CLI)
- Consulta anche

> **Note**
>
> Oltre al ruolo di servizio che autorizza l'accesso a livello di servizio, assegni un ruolo di scambio di token ai dispositivi principali di Greengrass. Il ruolo di scambio di token è un ruolo IAM separato che controlla il modo in cui i componenti Greengrass e le funzioni Lambda sul dispositivo principale possono accedere ai servizi. AWS Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

Gestire il ruolo del servizio Greengrass (console)

La console AWS IoT semplifica la gestione del ruolo del servizio Greengrass. Ad esempio, quando configuri il rilevamento dei dispositivi client per un dispositivo principale, la console verifica se il tuo Account AWS è collegato a un ruolo di servizio Greengrass nell'attuale. Regione AWS In caso contrario, la console può creare e configurare un ruolo del servizio automaticamente. Per ulteriori informazioni, consulta the section called “Creazione del ruolo del servizio Greengrass”.

È possibile utilizzare la console per le seguenti attività di gestione dei ruoli:

Argomenti

- Individuazione del ruolo del servizio Greengrass (console)
- Creazione del ruolo del servizio Greengrass (console)
- Modifica del ruolo del servizio Greengrass (console)
- Scollegare il ruolo del servizio Greengrass (console)

> **Note**
>
> L'utente che ha effettuato l'accesso alla console deve disporre delle autorizzazioni per visualizzare, creare o modificare il ruolo del servizio.

Individuazione del ruolo del servizio Greengrass (console)

Utilizza i seguenti passaggi per trovare il ruolo di servizio AWS IoT Greengrass utilizzato nella versione correnteRegione AWS.

1. Passare alla console AWS IoT.

2. Nel pannello di navigazione scegli Impostazioni.
3. Scorrere fino alla sezione Greengrass service role (Ruolo del servizio Greengrass) per visualizzare il ruolo del servizio e le relative policy.

   Se non vedi un ruolo di servizio, la console può crearne o configurarne uno per te. Per ulteriori informazioni, consulta Creazione del ruolo del servizio Greengrass.

Creazione del ruolo del servizio Greengrass (console)

La console può creare e configurare automaticamente un ruolo di servizio Greengrass predefinito. Il ruolo ha le proprietà seguenti:

Proprietà	Valore
Nome	`Greengrass_ServiceRole`
Trusted entity (Entità attendibile)	`AWS service: greengrass`
Policy	AWSGreengrassResourceAccessRolePolicy

> **Note**
>
> Se crei questo ruolo con lo script di configurazione del AWS IoT Greengrass V1 dispositivo, il nome del ruolo è`GreengrassServiceRole_`*`random-string`*.

Quando configuri il rilevamento dei dispositivi client per un dispositivo principale, la console verifica se un ruolo del servizio Greengrass è associato al tuo ruolo Account AWS nel sistema corrente. Regione AWS In caso contrario, la console richiede all'utente di consentire la lettura e AWS IoT Greengrass la scrittura sui AWS servizi per conto dell'utente.

Se concedi l'autorizzazione, la console verifica se `Greengrass_ServiceRole` esiste un ruolo denominato nel tuo. Account AWS

- Se il ruolo esiste, la console assegna il ruolo di servizio al tuo Account AWS ruolo attualeRegione AWS.
- Se il ruolo non esiste, la console crea un ruolo di servizio Greengrass predefinito e lo associa a quello correnteAccount AWS. Regione AWS

> **Note**
>
> Se desideri creare un ruolo di servizio con politiche di ruolo personalizzate, utilizza la console IAM per creare o modificare il ruolo. Per ulteriori informazioni, consulta Creazione di un ruolo per delegare le autorizzazioni a un AWS servizio o Modifica di un ruolo nella Guida per l'utente IAM. Assicurati che il ruolo conceda autorizzazioni equivalenti alle policy `AWSGreengrassResourceAccessRolePolicy` gestite per le funzionalità e le risorse utilizzate. Ti consigliamo di includere anche le chiavi di contesto `aws:SourceArn` e della condizione `aws:SourceAccount` globale nella tua politica di fiducia per evitare il confuso problema della sicurezza dei vicedirettori. Le chiavi di contesto delle condizioni limitano l'accesso per consentire solo le richieste che provengono dall'account specificato e dall'area di lavoro Greengrass. Per ulteriori informazioni sul problema del "confused deputy", consulta Prevenzione del problema "confused deputy" tra servizi.
> Se crei un ruolo di servizio, torna alla AWS IoT console e assegna il ruolo al tuo. Account AWS È possibile eseguire questa operazione nel ruolo di servizio Greengrass nella pagina Impostazioni.

Modifica del ruolo del servizio Greengrass (console)

Usa la seguente procedura per scegliere un ruolo di servizio Greengrass diverso da assegnare al tuo Account AWS nel ruolo Regione AWS attualmente selezionato nella console.

1. Passare alla console AWS IoT.
2. Nel pannello di navigazione scegli Impostazioni.
3. In Ruolo di servizio Greengrass, scegli Cambia ruolo.

   Si apre la finestra di dialogo Aggiorna ruolo di servizio Greengrass e mostra i ruoli IAM del tuo Account AWS che definisci AWS IoT Greengrass come entità attendibile.
4. Scegli il ruolo di servizio Greengrass da associare.
5. Scegli Allega ruolo.

Scollegare il ruolo del servizio Greengrass (console)

Utilizza la seguente procedura per scollegare il ruolo di servizio Greengrass dal AWS tuo account nella versione corrente. Regione AWS Ciò revoca le autorizzazioni per l'accesso AWS IoT Greengrass ai servizi AWS nella versione corrente. Regione AWS

> **Important**
>
> Lo scollegamento del ruolo del servizio potrebbe interrompere le operazioni attive.

1. Passare alla console AWS IoT.
2. Nel pannello di navigazione scegli Impostazioni.
3. Nel ruolo di servizio Greengrass, scegli il ruolo Detach.
4. Nella finestra di dialogo di conferma, scegli Detach (Scollega).

> **Note**
>
> Se non hai più bisogno del ruolo, puoi eliminarlo nella console IAM. Per ulteriori informazioni, consulta la sezione Eliminazione di ruoli o profili delle istanze nella Guida per l'utente di IAM. Altri ruoli potrebbero consentire ad AWS IoT Greengrass di accedere alle risorse. Per trovare tutti i ruoli che consentono di AWS IoT Greengrass assumere autorizzazioni per tuo conto, nella console IAM, nella pagina Ruoli, cerca i ruoli che includono AWSservice: greengrass nella colonna Trusted entities.

Gestione del ruolo di servizio Greengrass (CLI)

Nelle procedure seguenti, si presume che AWS Command Line Interface sia installato e configurato per utilizzare il tuo. Account AWS Per ulteriori informazioni, vedere Installazione, aggiornamento e disinstallazione AWS CLI e configurazione di AWS CLI nella Guida per l'AWS Command Line Interfaceutente.

Puoi utilizzare l'AWS CLI per le seguenti attività di gestione dei ruoli:

Argomenti

- Ottenimento del ruolo del servizio Greengrass (CLI)
- Creazione del ruolo del servizio Greengrass (CLI)
- Rimozione del ruolo del servizio Greengrass (CLI)

Ottenimento del ruolo del servizio Greengrass (CLI)

Utilizza la seguente procedura per scoprire se un ruolo di servizio Greengrass è associato al tuo Account AWS in un. Regione AWS

- Come ottenere il ruolo del servizio. Sostituisci la *regione* con la tua Regione AWS (ad esempio,`us-west-2`).

  ```
  aws greengrassv2 get-service-role-for-account --region region
  ```

  Se un ruolo di servizio Greengrass è già associato al tuo account, la richiesta restituisce i seguenti metadati del ruolo.

  ```
  {
    "associatedAt": "timestamp",
    "roleArn": "arn:aws:iam::account-id:role/path/role-name"
  }
  ```

  Se la richiesta non restituisce i metadati del ruolo, devi creare il ruolo di servizio (se non esiste) e associarlo al tuo account nel. Regione AWS

Creazione del ruolo del servizio Greengrass (CLI)

Utilizza i seguenti passaggi per creare un ruolo e associarlo al tuoAccount AWS.

Per creare il ruolo di servizio utilizzando IAM

1. Creare un ruolo con una policy di attendibilità che consenta a AWS IoT Greengrass di assumere tale ruolo. In questo esempio viene creato un ruolo denominato `Greengrass_ServiceRole`, ma è possibile utilizzare un nome diverso. Ti consigliamo di includere anche le chiavi del contesto `aws:SourceArn` e della condizione `aws:SourceAccount` globale nella tua politica di fiducia per evitare il confuso problema della vice sicurezza. Le chiavi di contesto delle condizioni limitano l'accesso per consentire solo le richieste che provengono dall'account specificato e dall'area di lavoro Greengrass. Per ulteriori informazioni sul problema del "confused deputy", consulta Prevenzione del problema "confused deputy" tra servizi.

Linux or Unix

```
aws iam create-role --role-name Greengrass_ServiceRole --assume-role-policy-
document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "greengrass.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:greengrass:region:account-id:*"
        },
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        }
      }
    }
  ]
}'
```

Windows Command Prompt (CMD)

```
aws iam create-role --role-name Greengrass_ServiceRole --assume-role-
policy-document "{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect
\":\"Allow\",\"Principal\":{\"Service\":\"greengrass.amazonaws.com\"},
\"Action\":\"sts:AssumeRole\",\"Condition\":{\"ArnLike\":{\"aws:SourceArn
\":\"arn:aws:greengrass:region:account-id:*\"},\"StringEquals\":
{\"aws:SourceAccount\":\"account-id\"}}}]}"
```

PowerShell

```
aws iam create-role --role-name Greengrass_ServiceRole --assume-role-policy-
document '{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "Service": "greengrass.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:greengrass:region:account-id:*"
        },
        "StringEquals": {
          "aws:SourceAccount": "account-id"
        }
      }
    }
  ]
}'
```

2. Copiare il ruolo ARN dai metadati del ruolo nell'output. Utilizzare l'ARN per associare un ruolo all'account.
3. Collegare la policy `AWSGreengrassResourceAccessRolePolicy` al ruolo.

```
aws iam attach-role-policy --role-name Greengrass_ServiceRole --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSGreengrassResourceAccessRolePolicy
```

Per associare il ruolo di servizio al tuo Account AWS

- Associare il ruolo all'account. Sostituisci *`role-arn`* con l'ARN del ruolo di servizio e la *`regione`* con il tuo Regione AWS (ad esempio,). `us-west-2`

```
aws greengrassv2 associate-service-role-to-account --role-arn role-arn --
region region
```

In caso di successo, la richiesta restituisce la seguente risposta.

```
{
  "associatedAt": "timestamp"
}
```

Rimozione del ruolo del servizio Greengrass (CLI)

Utilizza i seguenti passaggi per dissociare il ruolo di servizio Greengrass dal tuo. Account AWS

- Disassociare un ruolo del servizio dall'account. Sostituisci la *regione* con la tua Regione AWS (ad esempio,`us-west-2`).

  ```
  aws greengrassv2 disassociate-service-role-from-account --region region
  ```

  Se l'operazione riesce, viene restituita la seguente risposta.

  ```
  {
    "disassociatedAt": "timestamp"
  }
  ```

  > **Note**
  >
  > È necessario eliminare il ruolo di servizio se non lo si utilizza in nessunoRegione AWS. Per prima cosa utilizzare delete-role-policy per scollegare la policy gestita `AWSGreengrassResourceAccessRolePolicy` dal ruolo, quindi usare delete-role per eliminare il ruolo. Per ulteriori informazioni, consulta la sezione Eliminazione di ruoli o profili delle istanze nella Guida per l'utente di IAM.

Consulta anche

- Creazione di un ruolo per delegare le autorizzazioni a un AWS servizio nella IAM User Guide
- Modifica di un ruolo nella Guida per l'utente IAM
- Eliminazione di ruoli o profili di istanza nella IAM User Guide
- AWS IoT Greengrasscomandi nel AWS CLICommand Reference
  - associate-service-role-to-account
  - disassociate-service-role-from-conto
  - get-service-role-for-conto
- Comandi IAM nel AWS CLICommand Reference
  - attach-role-policy
  - create-role

- delete-role
- delete-role-policy

AWS Policy gestite da per AWS IoT Greengrass

Una policy gestita da AWS è una policy autonoma creata e amministrata da AWS. Le policy gestite da AWS sono progettate per fornire autorizzazioni per molti casi d'uso comuni in modo da poter iniziare ad assegnare autorizzazioni a utenti, gruppi e ruoli.

Ricorda che le policy gestite da AWS potrebbero non concedere autorizzazioni con privilegi minimi per i tuoi casi d'uso specifici perché possono essere utilizzate da tutti i clienti AWS. Consigliamo pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal cliente specifiche per i tuoi casi d'uso.

Non è possibile modificare le autorizzazioni definite nelle policy gestite da AWS. Se AWS aggiorna le autorizzazioni definite in una policy gestita da AWS, l'aggiornamento riguarda tutte le identità principali (utenti, gruppi e ruoli) a cui è collegata la policy. È molto probabile che AWS aggiorni una policy gestita da AWS quando viene lanciato un nuovo Servizio AWS o nuove operazioni API diventano disponibili per i servizi esistenti.

Per ulteriori informazioni, consultare Policy gestite da AWS nella Guida per l'utente di IAM.

Argomenti

- AWSPolicy gestita: AWSGreengrassFullAccess
- AWSPolicy gestita: AWSGreengrassReadOnlyAccess
- AWSPolicy gestita: AWSGreengrassResourceAccessRolePolicy
- Aggiornamenti di AWS IoT Greengrass alle policy gestite da AWS

AWSPolicy gestita: AWSGreengrassFullAccess

È possibile allegare la policy `AWSGreengrassFullAccess` alle identità IAM.

Questa politica concede autorizzazioni amministrative che consentono a un principale accesso completo a tuttiAWS IoT Greengrassazioni.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `greengrass`— Consente ai dirigenti l'accesso completo a tuttiAWS IoT Greengrassazioni.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:*"
            ],
            "Resource": "*"
        }
    ]
}
```

AWSPolicy gestita: AWSGreengrassReadOnlyAccess

È possibile allegare la policy `AWSGreengrassReadOnlyAccess` alle identità IAM.

Questa politica concede autorizzazioni di sola lettura che consentono a un preside di visualizzare, ma non modificare, le informazioni inAWS IoT Greengrass. Ad esempio, i responsabili con queste autorizzazioni possono visualizzare l'elenco dei componenti distribuiti su un dispositivo principale Greengrass, ma non possono creare una distribuzione per modificare i componenti in esecuzione su quel dispositivo.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `greengrass`— Consente ai committenti di eseguire azioni che restituiscono un elenco di elementi o dettagli su un articolo. Ciò include le operazioni API che iniziano con`ListoGet`.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "greengrass:List*",
                "greengrass:Get*"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

AWSPolicy gestita: AWSGreengrassResourceAccessRolePolicy

Puoi allegare`AWSGreengrassResourceAccessRolePolicy`policy per le tue entità IAM.AWS IoT Greengrassassocia questa politica anche a un ruolo di servizio che consenteAWS IoT Greengrassper eseguire azioni per tuo conto. Per ulteriori informazioni, consulta Ruolo del servizio Greengrass.

Questa politica concede autorizzazioni amministrative che consentonoAWS IoT Greengrassper eseguire attività essenziali, come il recupero delle funzioni Lambda, la gestioneAWS IoTombre sui dispositivi e verifica dei dispositivi client Greengrass.

Dettagli dell'autorizzazione

Questa policy include le seguenti autorizzazioni:

- `greengrass`— Gestisci le risorse Greengrass.
- `iot(*Shadow)` — GestisciAWS IoTombre che hanno i seguenti identificatori speciali nei loro nomi. Queste autorizzazioni sono necessarie affinchéAWS IoT Greengrasspuò comunicare con i dispositivi principali.
  - `*-gci`—AWS IoT Greengrassutilizza questa ombra per archiviare le informazioni di connettività principali dei dispositivi, in modo che i dispositivi client possano scoprire e connettersi ai dispositivi principali.
  - `*-gcm`—AWS IoT Greengrass V1utilizza questa ombra per notificare al dispositivo principale che il certificato dell'autorità di certificazione (CA) del gruppo Greengrass è stato ruotato.
  - `*-gda`—AWS IoT Greengrass V1utilizza questa ombra per notificare al dispositivo principale una distribuzione.
  - `GG_*`— Inutilizzato.
- `iot(DescribeThingeDescribeCertificate)` — Recupera informazioni suAWS IoTcose e certificati. Queste autorizzazioni sono necessarie affinchéAWS IoT Greengrasspuò verificare i dispositivi client che si connettono a un dispositivo principale. Per ulteriori informazioni, consulta Interagisci con dispositivi IoT locali.
- `lambda`— Recupera informazioni suAWS Lambdafunzioni. Questa autorizzazione è necessaria affinchéAWS IoT Greengrass V1può implementare funzioni Lambda sui core Greengrass. Per

ulteriori informazioni, vedereEsegui la funzione Lambda suAWS IoT Greengrasscentronel AWS IoT Greengrass V1Guida per gli sviluppatori.

- `secretsmanager`— Recupera il valore diAWS Secrets Managersegreti i cui nomi iniziano con`greengrass-`. Questa autorizzazione è necessaria affinchéAWS IoT Greengrass V1può distribuire i segreti di Secrets Manager ai core di Greengrass. Per ulteriori informazioni, vedereDistribuisci i segreti alAWS IoT Greengrasscentronel AWS IoT Greengrass V1Guida per gli sviluppatori.
- `s3`— Recupera file e oggetti dai bucket S3 i cui nomi contengono`greengrassosagemaker`. Queste autorizzazioni sono necessarie affinchéAWS IoT Greengrass V1può distribuire risorse di machine learning archiviate in bucket S3. Per ulteriori informazioni, vedereRisorse per l'apprendimento automaticonel AWS IoT Greengrass V1Guida per gli sviluppatori.
- `sagemaker`— Recupera informazioni su AmazonSageMakermodelli di inferenza di machine learning. Questa autorizzazione è necessaria affinchéAWS IoT Greengrass V1può distribuire modelli ML sui core Greengrass. Per ulteriori informazioni, vedereEsegui inferenze di apprendimento automaticonel AWS IoT Greengrass V1Guida per gli sviluppatori.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowGreengrassAccessToShadows",
            "Action": [
                "iot:DeleteThingShadow",
                "iot:GetThingShadow",
                "iot:UpdateThingShadow"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:iot:*:*:thing/GG_*",
                "arn:aws:iot:*:*:thing/*-gcm",
                "arn:aws:iot:*:*:thing/*-gda",
                "arn:aws:iot:*:*:thing/*-gci"
            ]
        },
        {
            "Sid": "AllowGreengrassToDescribeThings",
            "Action": [
                "iot:DescribeThing"
            ],
```

```
            "Effect": "Allow",
            "Resource": "arn:aws:iot:*:*:thing/*"
        },
        {
            "Sid": "AllowGreengrassToDescribeCertificates",
            "Action": [
                "iot:DescribeCertificate"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iot:*:*:cert/*"
        },
        {
            "Sid": "AllowGreengrassToCallGreengrassServices",
            "Action": [
                "greengrass:*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Sid": "AllowGreengrassToGetLambdaFunctions",
            "Action": [
                "lambda:GetFunction",
                "lambda:GetFunctionConfiguration"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Sid": "AllowGreengrassToGetGreengrassSecrets",
            "Action": [
                "secretsmanager:GetSecretValue"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:secretsmanager:*:*:secret:greengrass-*"
        },
        {
            "Sid": "AllowGreengrassAccessToS3Objects",
            "Action": [
                "s3:GetObject"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:s3:::*Greengrass*",
```

```
                "arn:aws:s3:::*GreenGrass*",
                "arn:aws:s3:::*greengrass*",
                "arn:aws:s3:::*Sagemaker*",
                "arn:aws:s3:::*SageMaker*",
                "arn:aws:s3:::*sagemaker*"
            ]
        },
        {
            "Sid": "AllowGreengrassAccessToS3BucketLocation",
            "Action": [
                "s3:GetBucketLocation"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Sid": "AllowGreengrassAccessToSageMakerTrainingJobs",
            "Action": [
                "sagemaker:DescribeTrainingJob"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:sagemaker:*:*:training-job/*"
            ]
        }
    ]
}
```

Aggiornamenti di AWS IoT Greengrass alle policy gestite da AWS

Puoi visualizzare i dettagli sugli aggiornamenti diAWSpolitiche gestite perAWS IoT Greengrassdal momento in cui questo servizio ha iniziato a tracciare queste modifiche. Per ricevere avvisi automatici sulle modifiche a questa pagina, iscriviti al feed RSS sulAWS IoT Greengrass V2pagina di cronologia del documento.

Modifica	Descrizione	Data
AWS IoT GreengrassRilevamento delle modifiche	AWS IoT Greengrass ha iniziato a monitorare le modifiche per le sue policy gestite da AWS.	2 luglio 2021

Prevenzione del problema "confused deputy" tra servizi

Con "confused deputy" si intende un problema di sicurezza in cui un'entità che non dispone dell'autorizzazione per eseguire una certa operazione può costringere un'entità con più privilegi a eseguire tale operazione. In AWS, la rappresentazione cross-service può comportare il problema confused deputy. La rappresentazione tra servizi può verificarsi quando un servizio (il servizio chiamante) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per utilizzare le proprie autorizzazioni e agire sulle risorse di un altro cliente, a cui normalmente non avrebbe accesso. Per evitare ciò, AWS fornisce strumenti per poterti a proteggere i tuoi dati per tutti i servizi con entità di servizio a cui è stato concesso l'accesso alle risorse del tuo account.

Ti consigliamo di utilizzare le chiavi di contesto delle condizioni globali `aws:SourceArn` e `aws:SourceAccount` nelle policy delle risorse per limitare le autorizzazioni con cui AWS IoT Greengrass fornisce un altro servizio alla risorsa. Se si utilizzano entrambe le chiavi di contesto delle condizioni globali, il valore `aws:SourceAccount` e l'account nel valore `aws:SourceArn` devono utilizzare lo stesso ID account nella stessa istruzione di policy.

Il valore di`aws:SourceArn`deve essere la risorsa cliente Greengrass associata al`sts:AssumeRole`.

Il modo più efficace per proteggersi dal problema "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa. Se non si conosce l'ARN completo della risorsa o si scelgono più risorse, è necessario utilizzare la chiave di contesto della condizione globale `aws:SourceArn` con caratteri jolly (`*`) per le parti sconosciute dell'ARN. Ad esempio, `arn:aws:greengrass::`*`account-id`*`:*`.

Per un esempio di policy che utilizza`aws:SourceArn`e`aws:SourceAccount`Chiavi di contesto delle condizioni globali, vediCreazione del ruolo del servizio Greengrass.

Risoluzione dei problemi di identità e accesso per AWS IoT Greengrass

Utilizza le informazioni seguenti per diagnosticare e risolvere i problemi comuni che possono verificarsi durante l'utilizzo di AWS IoT Greengrass e di IAM.

Problemi

- Non sono autorizzato a eseguire un'operazione in AWS IoT Greengrass
- Non sono autorizzato a eseguire iam:PassRole

- Sono un amministratore e desidero consentire ad altri utenti di accedere a AWS IoT Greengrass
- Voglio consentire alle persone esterne al mio account Account AWS di accedere alle mie risorse AWS IoT Greengrass

Per un aiuto generale nella risoluzione dei problemi, consulta Risoluzione dei problemi.

Non sono autorizzato a eseguire un'operazione in AWS IoT Greengrass

Se ricevi un errore che indica che non sei autorizzato a eseguire un'operazione, devi contattare il tuo amministratore per ricevere assistenza. L'amministratore è la persona che ti ha fornito il nome utente e la password.

L'errore di esempio seguente si verifica quando`mateojackson`L'utente IAM tenta di visualizzare i dettagli su un dispositivo principale, ma non dispone`greengrass:GetCoreDevice`autorizzazioni.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to
 perform: greengrass:GetCoreDevice on resource: arn:aws:greengrass:us-
west-2:123456789012:coreDevices/MyGreengrassCore
```

In questo caso, Mateo richiede al suo amministratore di aggiornare le sue policy per poter accedere alla risorsa `arn:aws:greengrass:us-west-2:123456789012:coreDevices/MyGreengrassCore` utilizzando l'operazione `greengrass:GetCoreDevice`.

Di seguito sono riportati i problemi generali di IAM che possono verificarsi durante l'utilizzo diAWS IoT Greengrass.

Non sono autorizzato a eseguire iam:PassRole

Se ricevi un errore che indica che non sei autorizzato a eseguire l'operazione `iam:PassRole`, le tue policy devono essere aggiornate per poter passare un ruolo a AWS IoT Greengrass.

Alcuni Servizi AWS consentono di passare un ruolo esistente a tale servizio, invece di creare un nuovo ruolo di servizio o un ruolo collegato ai servizi. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per passare il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione in AWS IoT Greengrass. Tuttavia, l'operazione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Per ulteriore assistenza con l'accesso, contatta l'amministratore AWS. L'amministratore è colui che ti ha fornito le credenziali di accesso.

Sono un amministratore e desidero consentire ad altri utenti di accedere a AWS IoT Greengrass

Per consentire ad altri utenti di accedere ad AWS IoT Greengrass, devi creare un'entità IAM (utente o ruolo) per la persona o l'applicazione che richiede l'accesso. Tale utente o applicazione utilizzerà le credenziali dell'entità per accedere ad AWS. Dovrai quindi collegare all'entità una policy che conceda le autorizzazioni corrette in AWS IoT Greengrass.

Per iniziare immediatamente, consulta Creazione dei primi utenti e gruppi delegati IAM nella Guida per l'utente di IAM.

Voglio consentire alle persone esterne al mio account Account AWS di accedere alle mie risorse AWS IoT Greengrass

Puoi creare un ruolo IAM che può essere utilizzato dagli utenti in altri account o da persone esterne all'organizzazione per accedere alAWSrisorse. Puoi specificare chi è attendibile per l'assunzione del ruolo. Per ulteriori informazioni, consulta la paginaFornire l'accesso a un utente IAM in un altroAccount AWSche possiedieFornire l'accesso aAccount AWSè di proprietà di terzinellIAM User Guide.

AWS IoT Greengrass non supporta l'accesso tra account basato su policy basate su risorse o liste di controllo accessi (ACL).

Consenti il traffico dei dispositivi tramite un proxy o un firewall

I dispositivi core Greengrass e i componenti Greengrass eseguono le richieste in uscita verso servizi e altri siti Web. AWS Come misura di sicurezza, potresti limitare il traffico in uscita a un piccolo intervallo di endpoint e porte. Puoi utilizzare le seguenti informazioni su endpoint e porte per limitare

il traffico dei dispositivi tramite un proxy, un firewall o un gruppo di sicurezza Amazon VPC. Per ulteriori informazioni su come configurare un dispositivo principale per l'utilizzo di un proxy, consulta. Connessione alla porta 443 o tramite un proxy di rete

Argomenti

Endpoint per il funzionamento di base

I dispositivi core Greengrass utilizzano i seguenti endpoint e porte per il funzionamento di base.

Recupera gli endpoint AWS IoT

Ottieni gli AWS IoT endpoint per te e salvali per Account AWS utilizzarli in un secondo momento. Il tuo dispositivo utilizza questi endpoint per connettersi a. AWS IoT Esegui questa operazione:

1. Ottieni l'endpoint di AWS IoT dati per il tuo. Account AWS

   ```
   aws iot describe-endpoint --endpoint-type iot:Data-ATS
   ```

   La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

   ```
   {
     "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
   }
   ```

2. Ottieni l'endpoint delle AWS IoT credenziali per il tuo. Account AWS

   ```
   aws iot describe-endpoint --endpoint-type iot:CredentialProvider
   ```

   La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

   ```
   {
     "endpointAddress": "device-credentials-prefix.credentials.iot.us-
   west-2.amazonaws.com"
   }
   ```

Endpoint	Porta	Richiesto	Descrizione
`greengrass-ats.iot. region.amazonaws.com`	8443 o 443	Sì	Utilizzato per operazioni sul piano dati, come l'installazione di distribuzioni e l'utilizzo di dispositivi client.
device-data-prefix `-ats.iot. region.amazonaws.com`	MQTT: 8883 o 443 HTTPS: 8443 o 443	Sì	Utilizzato per operazioni sul piano dati per la gestione dei dispositivi, come la comunicazione MQTT e la sincronizzazione delle ombre con. AWS IoT Core
device-credentials-prefix `.credenti`	443	Sì	Utilizzato per

Endpoint	Porta	Richiesto	Descrizione
`als.iot. region.amazonaws.com`			acquisire AWS credenziali, che il dispositivo principale utilizza per scaricare elementi dei componenti da Amazon S3 ed eseguire altre operazioni. Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

Endpoint	Porta	Richiesto	Descrizione

`*.s3.amazonaws.com`
 	
`*.s3.`*`region`*`.amazonaws.com`
	443	Sì	Utilizzato per le distribuzioni. Questo formato include il * carattere, poiché i prefissi degli endpoint sono controllati internamente e possono cambiare in qualsiasi momento.

Endpoint	Porta	Richiesto	Descrizione
`data.iot. region.amazonaws.com`	443	No	Richiesto se il dispositivo principale esegue una versione del nucleo di Greengrass precedente alla v2.4.0 ed è configurato per utilizzare un proxy di rete. Il dispositivo principale utilizza questo endpoint per la comunicazione MQTT con chi è protetto da un proxy. AWS IoT Core Per ulteriori informazi

Endpoint	Porta	Richiesto	Descrizione
			oni, consulta Configurare un proxy di rete.

Endpoint per l'installazione con provisioning automatico

I dispositivi core Greengrass utilizzano i seguenti endpoint e porte quando si installa il software AWS IoT Greengrass Core con provisioning automatico delle risorse.

Endpoint	Porta	Richiesto	Descrizione
`iot.region.amazonaws.com`	443	Sì	Utilizzato per creare AWS IoT risorse e recuperare informazioni sulle risorse esistenti. AWS IoT
`iam.amazonaws.com`	443	Sì	Utilizzato per creare risorse IAM e recuperare informazioni sulle risorse IAM esistenti.

Endpoint	Porta	Richiesto	Descrizione
`sts.region.amazonaws.com`	443	Sì	Utilizzato per ottenere l'ID del tuoAccount AWS.
`greengrass. region.amazonaws.com`	443	No	Obbligatorio se si utilizza l'`--deploy-dev-tools` argomento per distribuire il componente Greengrass CLI sul dispositivo principale.

Endpoint per i componenti forniti AWS

I dispositivi core Greengrass utilizzano endpoint aggiuntivi a seconda dei componenti software eseguiti. Puoi trovare gli endpoint richiesti da ciascun componente AWS fornito nella sezione Requisiti della pagina relativa a ciascun componente di questa guida per sviluppatori. Per ulteriori informazioni, consulta AWS-componenti forniti.

Convalida della conformità per AWS IoT Greengrass

Per sapere se il Servizio AWSè coperto da programmi di conformità specifici, consulta i Servizi AWScoperti dal programma di conformità e scegli il programma di conformità desiderato. Per informazioni generali, consulta Programmi per la conformità di AWS.

È possibile scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta Download di report in AWS Artifact.

La responsabilità di conformità durante l'utilizzo dei Servizi AWSè determinata dalla riservatezza dei dati, dagli obiettivi di conformità dell'azienda e dalle normative vigenti. Per semplificare il rispetto della conformità, AWSmette a disposizione le seguenti risorse:

- Guide Quick Start per la sicurezza e conformità: queste guide all'implementazione illustrano considerazioni relative all'architettura e forniscono la procedura per l'implementazione di ambienti di base su AWSincentrati sulla sicurezza e sulla conformità.
- Architetture per la sicurezza e la conformità HIPAA su Amazon Web Services: questo whitepaper descrive come le aziende possono utilizzare AWSper creare applicazioni conformi alla normativa HIPAA.

  > **Note**
  >
  > Non tutti i Servizi AWSsono conformi ai requisiti HIPAA. Per ulteriori informazioni, consulta la sezione Riferimenti sui servizi conformi ai requisiti HIPAA.

- Risorse per la conformità AWS: una raccolta di cartelle di lavoro e guide suddivise per settore e area geografica.
- AWSGuide alla conformità dei clienti: comprendi il modello di responsabilità condivisa attraverso la lente della conformità. Le guide riassumono le migliori pratiche per la protezione Servizi AWS e mappano le linee guida per i controlli di sicurezza su più framework (tra cui il National Institute of Standards and Technology (NIST), il Payment Card Industry Security Standards Council (PCI) e l'International Organization for Standardization (ISO)).
- Valutazione delle risorse con le regole nella Guida per gli sviluppatori di AWS Config: il servizio AWS Configvaluta il livello di conformità delle configurazioni delle risorse con pratiche interne, linee guida e regolamenti.
- AWS Security Hub: questo Servizio AWSfornisce una visione completa dello stato di sicurezza all'interno di AWS. La Centrale di sicurezza utilizza i controlli di sicurezza per valutare le risorse

AWSe verificare la conformità agli standard e alle best practice del settore della sicurezza. Per un elenco dei servizi e dei controlli supportati, consulta la pagina Documentazione di riferimento sui controlli della Centrale di sicurezza.

- AWS Audit Manager: questo Servizio AWSaiuta a verificare continuamente l'utilizzo di AWSper semplificare la gestione dei rischi e della conformità alle normative e agli standard di settore.

Resilienza in AWS IoT Greengrass

LaAWSL'infrastruttura globale di è basata su regioni e zone di disponibilità di Amazon Web Services. EACHRegione AWSLe Regioni forniscono più zone di disponibilità fisicamente separate e isolate che sono connesse tramite reti altamente ridondanti, a bassa latenza e throughput elevato. Con le Zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le Zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili, rispetto alle infrastrutture a data center singolo o multiplo.

Per ulteriori informazioni, consulta Infrastruttura globale di AWS.

Oltre all'infrastruttura globale di AWS, AWS IoT Greengrass offre numerose funzionalità per supportare la resilienza dei dati e le esigenze di backup.

- È possibile configurare un dispositivo core Greengrass per scrivere i registri nel file system locale e in CloudWatch registri. Se il dispositivo core perde la connettività, può continuare a registrare i messaggi nel file system. Quando si riconnette, scrive i messaggi di log su CloudWatch registri. Per ulteriori informazioni, consulta la pagina Monitora AWS IoT Greengrass i registri .
- Se un dispositivo core perde energia durante una distribuzione, riprende la distribuzione dopo laAWS IoT GreengrassIl software core riparte.
- Se un dispositivo core perde la connettività Internet, i dispositivi client Greengrass possono continuare a comunicare tramite la rete locale.
- È possibile creare componenti Greengrass che leggonoStream managertrasmette e inviano i dati a destinazioni di storage locali.

Sicurezza dell'infrastruttura in AWS IoT Greengrass

In qualità di servizio gestito, AWS IoT Greengrass è protetto dalle procedure di sicurezza di rete globali di AWS descritte nel whitepaper Amazon Web Services: Panoramica dei processi di sicurezza.

Utilizza le chiamate API pubblicate di AWS per accedere a AWS IoT Greengrass tramite la rete. I client devono supportare Transport Layer Security (TLS) 1.2 o versioni successive. È consigliabile TLS 1.3 o versioni successive. I client devono inoltre supportare le suite di cifratura con PFS (Perfect Forward Secrecy), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta che è associata a un'entità IAM. In alternativa, è possibile utilizzare AWS Security Token Service (AWS STS) per generare le credenziali di sicurezza temporanee per sottoscrivere le richieste.

In un AWS IoT Greengrass ambiente, i dispositivi utilizzano certificati X.509 e chiavi crittografiche per connettersi e autenticarsi a. Cloud AWS Per ulteriori informazioni, consulta the section called “Autenticazione e autorizzazione del dispositivo”.

Analisi della configurazione e delle vulnerabilità in AWS IoT Greengrass

Gli ambienti IoT possono essere costituiti da un numero elevato di dispositivi con funzionalità diverse, usati per lunghi periodi di tempo e distribuiti in varie aree geografiche. Queste caratteristiche rendono la configurazione di un dispositivo complessa e soggetta a errori. E poiché i dispositivi presentano spesso vincoli di potenza di elaborazione, memoria e capacità di storage, ciò limita l'uso della crittografia e di altre forme di sicurezza nei dispositivi stessi. I dispositivi, inoltre, usano spesso software con vulnerabilità note. La combinazione di questi fattori rende i dispositivi IoT un facile bersaglio per gli hacker e ne rende difficile la protezione continuativa

AWS IoT Device Defender risolve queste sfide fornendo strumenti per identificare i problemi di sicurezza e il mancato rispetto delle best practice. È possibile utilizzare AWS IoT Device Defender per analizzare, controllare e monitorare i dispositivi connessi per rilevare comportamenti anomali e ridurre i rischi per la sicurezza. AWS IoT Device Defender può controllare i dispositivi per assicurarsi che rispettino le best practice per la sicurezza e rilevare comportamenti anomali sui dispositivi. Offre la possibilità di applicare policy di sicurezza coerenti in tutti i dispositivi e di rispondere rapidamente quando i dispositivi vengono compromessi. Per ulteriori informazioni, vedi i seguenti argomenti:

- LaComponente di Device Defender
- AWS IoT Device Defender nella Guida per gli sviluppatori di AWS IoT Core.

Negli ambienti AWS IoT Greengrass, è necessario essere consapevoli delle seguenti considerazioni:

- È propria responsabilità proteggere i dispositivi fisici, il file system sui dispositivi e la rete locale.
- AWS IoT Greengrassnon impone l'isolamento di rete per i componenti Greengrass definiti dall'utente, indipendentemente dal fatto che siano eseguite o meno in un Container Greengrass. Pertanto, è possibile per i componenti Greengrass comunicare con qualsiasi altro processo in esecuzione nel sistema o fuori rete.

Integrità del codice inAWS IoT Greengrass V2

AWS IoT Greengrassdistribuisce componenti software dalCloud AWSai dispositivi che eseguono ilAWS IoT GreengrassSoftware Core. Questi componenti software includono:AWS-componenti fornitiecomponenti personalizzatiche carichi sul tuoAccount AWS. Ogni componente è composto da una ricetta. La ricetta definisce i metadati del componente e qualsiasi numero di artefatti, che sono componenti binari, come codice compilato e risorse statiche. Gli artefatti dei componenti sono archiviati in Amazon S3.

Man mano che sviluppi e distribuisci componenti Greengrass, segui questi passaggi fondamentali che funzionano con gli artefatti dei componenti nel tuoAccount AWSe sui tuoi dispositivi:

1. Crea e carica artefatti su bucket S3.
2. Creare un componente da una ricetta e artefatti nellaAWS IoT Greengrassservizio, che calcola unhash crittograficodi ogni artefatto.
3. Distribuisci un componente sui dispositivi core Greengrass, che scaricano e verificano l'integrità di ciascun artefatto.

AWSè responsabile del mantenimento dell'integrità degli artefatti dopo aver caricato gli artifact nei bucket S3, anche quando si distribuiscono componenti sui dispositivi core Greengrass. L'utente è responsabile della protezione degli artefatti del software prima di caricare gli artefatti in bucket S3. Sei inoltre responsabile della sicurezza dell'accesso alle risorse presenti nel tuoAccount AWS, inclusi i bucket S3 in cui si caricano gli artefatti dei componenti.

> **Note**
>
> Amazon S3 fornisce una funzione chiamata S3 Object Lock che puoi utilizzare per proteggere dalle modifiche agli artefatti dei componenti nei bucket S3Account AWS. Puoi utilizzarlo per impedire che gli artifact dei componenti vengano eliminati o sovrascritti. Per ulteriori

informazioni, consultaUtilizzo del blocco oggetti S3nellaGuida utente Amazon Simple Storage Service.

QuandoAWSpubblica un componente pubblico e quando carichi un componente personalizzato,AWS IoT Greengrasscalcola un digest crittografico per ciascun artefatto componente.AWS IoT Greengrassaggiorna la ricetta del componente per includere il digest di ciascun artefatto e l'algoritmo hash utilizzato per calcolare quel digest. Questo digest garantisce l'integrità dell'artefatto, perché se l'artefatto cambia nellaCloud AWSo durante il download, il file digest non corrisponderà al digestAWS IoT Greengrassmemorizza nella ricetta del componente. Per ulteriori informazioni, consultaArtefatti nel riferimento della ricetta del componente.

Quando si distribuisce un componente su un dispositivo core,AWS IoT GreengrassIl software di base scarica la ricetta del componente e ogni elemento che la ricetta definisce. LaAWS IoT GreengrassIl software Core calcola il digest di ciascun file di artefatto scaricato e lo confronta con il digest di quell'artefatto nella ricetta. In caso contrario, la distribuzione non riesce eAWS IoT GreengrassIl software principale elimina gli artefatti scaricati dal file system del dispositivo. Per ulteriori informazioni su come connette i dispositivi core eAWS IoT Greengrasssono protetti, vediCrittografia dei dati in transito.

L'utente è responsabile della protezione dei file degli artifact dei componenti sui file system dei dispositivi principali. LaAWS IoT GreengrassIl software core salva gli artefatti nel`packages`cartella nella cartella principale di Greengrass. É possibile utilizzareAWS IoT Device Defenderper analizzare, controllare e monitorare i dispositivi principali. Per ulteriori informazioni, consultare Analisi della configurazione e delle vulnerabilità in AWS IoT Greengrass.

AWS IoT Greengrass ed endpoint VPC dell'interfaccia (AWS PrivateLink)

Puoi stabilire una connessione privata tra il tuo VPC e il piano di AWS IoT Greengrass controllo creando un endpoint VPC di interfaccia. È possibile utilizzare questo endpoint per gestire componenti, implementazioni e dispositivi principali del servizio. AWS IoT Greengrass Gli endpoint di interfaccia sono alimentati da AWS PrivateLink, una tecnologia che consente di accedere alle AWS IoT Greengrass API in modo privato senza un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione Direct ConnectAWS. Le istanze presenti nel VPC non richiedono indirizzi IP pubblici per comunicare con le API dell'AWS IoT Greengrass. Il traffico tra il tuo VPC e l'AWS IoT Greengrass non esce dalla rete Amazon.

Ogni endpoint dell'interfaccia è rappresentato da una o più interfacce di rete elastiche nelle tue sottoreti.

Per ulteriori informazioni, consultare Endpoint VPC di interfaccia (AWS PrivateLink) nella Guida per l'utente di Amazon VPC.

Argomenti

- Considerazioni sugli endpoint VPC dell'AWS IoT Greengrass
- Crea un endpoint VPC di interfaccia per AWS IoT Greengrass le operazioni del piano di controllo
- Creazione di una policy di endpoint VPC per l'AWS IoT Greengrass
- Gestisci un dispositivo AWS IoT Greengrass principale in VPC

Considerazioni sugli endpoint VPC dell'AWS IoT Greengrass

Prima di configurare un endpoint VPC di interfaccia perAWS IoT Greengrass, consulta le proprietà e le limitazioni dell'endpoint dell'interfaccia nella Amazon VPC User Guide. Inoltre, tieni presente le seguenti considerazioni:

- AWS IoT Greengrasssupporta l'esecuzione di chiamate a tutte le azioni dell'API del piano di controllo dal tuo VPC. Il piano di controllo include operazioni come CreateDeploymente ListEffectiveDeployments. Il piano di controllo non include operazioni come ResolveComponentCandidatese Discover, che sono operazioni sul piano dati.
- Gli endpoint VPC per non AWS IoT Greengrass sono attualmente supportati nelle AWS regioni della Cina.

Crea un endpoint VPC di interfaccia per AWS IoT Greengrass le operazioni del piano di controllo

Puoi creare un endpoint VPC per il piano di AWS IoT Greengrass controllo utilizzando la console Amazon VPC o (). AWS Command Line Interface AWS CLI Per ulteriori informazioni, consulta Creazione di un endpoint di interfaccia nella Guida per l'utente di Amazon VPC.

Crea un endpoint VPC per AWS IoT Greengrass, utilizzando il seguente nome di servizio:

- com.amazonaws.*region*.greengrass

Se si abilita il DNS privato per l'endpoint, è possibile effettuare richieste API verso AWS IoT Greengrassutilizzando il nome DNS predefinito per la regione, ad esempio `greengrass.us-east-1.amazonaws.com`. DNS privato è abilitato per impostazione predefinita.

Per ulteriori informazioni, consulta Accesso a un servizio tramite un endpoint dell'interfaccia in Guida per l'utente di Amazon VPC.

Creazione di una policy di endpoint VPC per l'AWS IoT Greengrass

Puoi allegare una policy per gli endpoint all'endpoint VPC che controlla l'accesso AWS IoT Greengrass alle operazioni del piano di controllo. La policy specifica le informazioni riportate di seguito:

- Il principale che può eseguire operazioni.
- Le azioni che l'entità può eseguire.
- Le risorse su cui il preside può eseguire azioni.

Per ulteriori informazioni, consulta Controllo degli accessi ai servizi con endpoint VPC nella Guida per l'utente di Amazon VPC.

Example Esempio: policy di endpoint VPC per le operazioni dell'AWS IoT Greengrass

Di seguito è riportato un esempio di una policy endpoint per l'AWS IoT Greengrass. Se collegato a un endpoint, questo criterio concede l'accesso alle operazioni dell'AWS IoT Greengrasselencate per tutti gli account principali su tutte le risorse.

```
{
    "Statement": [
        {
            "Principal": "*",
            "Effect": "Allow",
            "Action": [
                "greengrass:CreateDeployment",
                "greengrass:ListEffectiveDeployments"
            ],
            "Resource": "*"
        }
    ]
}
```

Gestisci un dispositivo AWS IoT Greengrass principale in VPC

Puoi utilizzare un dispositivo core Greengrass ed eseguire implementazioni in VPC senza accesso pubblico a Internet. Come minimo, è necessario configurare i seguenti endpoint VPC con gli alias DNS corrispondenti. Per ulteriori informazioni su come creare e utilizzare endpoint VPC, consulta Create a VPC endpoint nella Amazon VPC User Guide.

> **Note**
>
> La funzionalità VPC per la creazione automatica di un record DNS è disattivata per AWS IoT data le credenziali e. AWS IoT Per connettere questi endpoint, è necessario creare manualmente un record DNS privato. Per ulteriori informazioni, consulta DNS privato per endpoint di interfaccia. Per ulteriori informazioni sulle limitazioni del AWS IoT Core VPC, consulta Limitazioni degli endpoint VPC.

Prerequisiti

- È necessario installare il software AWS IoT Greengrass Core utilizzando le fasi di provisioning manuale. Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning manuale delle risorse.

Limitazioni

- L'utilizzo di un dispositivo core Greengrass in VPC non è supportato nelle regioni cinesi e. AWS GovCloud (US) Regions
- Per ulteriori informazioni sulle limitazioni AWS IoT data e sugli endpoint VPC del provider di AWS IoT credenziali, consulta Limitazioni.

Configura il tuo dispositivo principale Greengrass per funzionare in VPC

1. Ottieni gli AWS IoT endpoint per te e salvali per Account AWS utilizzarli in un secondo momento. Il tuo dispositivo utilizza questi endpoint per connettersi a. AWS IoT Esegui questa operazione:

   a. Ottieni l'endpoint di AWS IoT dati per il tuo. Account AWS

```
aws iot describe-endpoint --endpoint-type iot:Data-ATS
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "endpointAddress": "device-data-prefix-ats.iot.us-west-2.amazonaws.com"
}
```

b. Ottieni l'endpoint delle AWS IoT credenziali per il tuo. Account AWS

```
aws iot describe-endpoint --endpoint-type iot:CredentialProvider
```

La risposta è simile all'esempio seguente, se la richiesta ha esito positivo.

```
{
  "endpointAddress": "device-credentials-prefix.credentials.iot.us-
west-2.amazonaws.com"
}
```

2. Crea un'interfaccia Amazon VPC per gli endpoint AWS IoT data e AWS IoT le credenziali:

   a. Vai alla console VPC Endpoints, sotto Virtual private cloud nel menu a sinistra, scegli Endpoints quindi Crea endpoint.

   b. Nella pagina Crea endpoint, specifica le seguenti informazioni.

   - Scegli Servizio AWSs per Categoria di servizio.
   - Per Service Name (Nome servizio), esegui la ricerca inserendo la parola chiave `iot`. Nell'elenco dei `iot` servizi visualizzati, scegli l'endpoint.

     Se crei un endpoint VPC per il piano AWS IoT Core dati, scegli l'endpoint API del piano AWS IoT Core dati per la tua regione. L'endpoint sarà del formato `com.amazonaws.region.iot.data`.

     Se crei un endpoint VPC per un provider di AWS IoT Core credenziali, scegli l'endpoint del provider di AWS IoT Core credenziali per la tua regione. L'endpoint sarà del formato `com.amazonaws.region.iot.credentials`.

> **Note**
>
> Il nome del servizio per il piano AWS IoT Core dati nella regione cinese avrà il seguente formato. `cn.com.amazonaws.region.iot.data` La creazione di endpoint VPC per il provider di AWS IoT Core credenziali non è supportata nella regione Cina.

- Per VPC e Subnets (Sottoreti) scegli il VPC in cui desideri creare l'endpoint e le zone di disponibilità in cui desideri creare la rete endpoint.
- Per Enable DNS name (Abilitare nome DNS), assicurati che Enable for this endpoint (Abilita per questo endpoint) non sia selezionata. Né il piano AWS IoT Core dati né il provider di AWS IoT Core credenziali supportano ancora i nomi DNS privati.
- In Security group (Gruppo di sicurezza), scegli i gruppi di sicurezza da associare alle interfacce di rete dell'endpoint.
- Facoltativamente, puoi aggiungere o rimuovere i tag. I tag sono coppie nome-valore utilizzate per l'associazione al tuo endpoint.

c. Per creare l'endpoint VPC, scegli Create endpoint (Crea endpoint).

3. Dopo aver creato l'AWS PrivateLinkendpoint, nella scheda Dettagli dell'endpoint, vedrai un elenco di nomi DNS. Puoi utilizzare uno di questi nomi DNS che hai creato in questa sezione per configurare la tua zona ospitata privata.
4. Crea un endpoint Amazon S3. Per ulteriori informazioni, consulta Creare un endpoint VPC per Amazon S3.
5. Se si utilizzano componenti Greengrass AWS forniti, potrebbero essere necessari endpoint e configurazioni aggiuntivi. Per visualizzare i requisiti degli endpoint, seleziona il componente dall'elenco dei componenti AWS forniti e consulta la sezione Requisiti. Ad esempio, i requisiti del componente log manager suggeriscono che questo componente deve essere in grado di eseguire richieste in uscita verso l'endpoint. `logs.region.amazonaws.com`

   Se si utilizza un componente personalizzato, potrebbe essere necessario esaminare le dipendenze ed eseguire ulteriori test per determinare se sono necessari endpoint aggiuntivi.
6. Nella configurazione Greengrass nucleus, `greengrassDataPlaneEndpoint` deve essere impostato su. **`iotdata`** Per ulteriori informazioni, vedere Greengrass nucleus configuration.

7. Se ti trovi nella `us-east-1` regione, imposta il parametro di configurazione `s3EndpointType` su **`REGIONAL`** nella configurazione Greengrass nucleus. Questa funzionalità è disponibile per le versioni 2.11.3 o successive di Greengrass nucleus.

Example Esempio: configurazione dei componenti

```
{
"aws.greengrass.Nucleus": {
   "configuration": {
      "awsRegion": "us-east-1",
      "iotCredEndpoint": "xxxxxx.credentials.iot.region.amazonaws.com",
      "iotDataEndpoint": "xxxxxx-ats.iot.region.amazonaws.com",
      "greengrassDataPlaneEndpoint": "iotdata",
      "s3EndpointType": "REGIONAL"
      ...
     }
   }
}
```

La tabella seguente fornisce informazioni sugli alias DNS privati personalizzati corrispondenti.

Servizio	Nome del servizio endpoint VPC	Tipo di endpoint VPC	Alias DNS privato personalizzato	Note
AWS IoT data	`com.amazonaws.` *`region`*`.iot.d`	Interfaccia	*`prefix`*`-ats.iot.` *`region`*`.a` `s.com`	Il record DNS privato deve corrispondere all'endpoint del AWS IoT data tuo account:.

Servizio	Nome del servizio endpoint VPC	Tipo di endpoint VPC	Alias DNS privato personalizzato	Note
				`aws iot describe-endpoint --endpoint-type iot:Data-ATS`
AWS IoTCredenziali	`com.amazonaws.` *`region`*`.iot.credentials`	Interfaccia	*`prefix`*`.credentials.iot.us.com`	Il record DNS privato deve corrispondere all'endpoint delle AWS IoT credenziali del tuo account:. `aws iot describe-endpoint --endpoint-type iot:CredentialProvider`

Servizio	Nome del servizio endpoint VPC	Tipo di endpoint VPC	Alias DNS privato personalizzato	Note
Amazon S3	`com.amazonaws.` *`region`*`.s3`	Interfaccia		Il record DNS viene creato automaticamente.

Best practice relative alla sicurezza di AWS IoT Greengrass

In questo argomento sono contenute le best practice per la sicurezza di AWS IoT Greengrass.

Concedere autorizzazioni minime possibili

Segui il principio del privilegio minimo per i tuoi componenti eseguendoli come utenti senza privilegi. I componenti non devono essere eseguiti come root a meno che non sia assolutamente necessario.

Utilizza il set minimo di autorizzazioni nei ruoli IAM. Limita l'uso di*jolly per`Action`e`Resource`proprietà nelle tue politiche IAM. Invece, dichiarare un insieme finito di operazioni e risorse quando possibile. Per ulteriori informazioni su privilegi minimi e altre best practice sulle policy, consulta the section called “Best practice per le policy”.

La best practice con privilegi minimi si applica anche aAWS IoTle politiche che allegate al vostro core Greengrass.

Non codificate le credenziali nei componenti Greengrass

Non codificate le credenziali nei componenti Greengrass definiti dall'utente. Per proteggere meglio le credenziali:

- Per interagire conAWSservizi, definisci le autorizzazioni per azioni e risorse specifiche inRuolo di servizio principale di Greengrass per i dispositivi.

- Usa ilcomponente gestore segretoper memorizzare le tue credenziali. Oppure, se la funzione utilizzaAWSSDK, utilizza le credenziali della catena di provider di credenziali predefinita.

Non registrare informazioni riservate

È necessario impedire la registrazione delle credenziali e di altre informazioni di identificazione personale (PII). Ti consigliamo di implementare le seguenti misure di protezione anche se l'accesso ai log locali su un dispositivo principale richiede i privilegi di root e l'accesso a CloudWatch I log richiedono autorizzazioni IAM.

- Non utilizzare informazioni riservate nei percorsi argomento MQTT.
- Non utilizzare informazioni riservate nei nomi, nei tipi e negli attributi dei dispositivi nel Registro di sistema AWS IoT Core.
- Non registrate informazioni sensibili nei componenti Greengrass definiti dall'utente o nelle funzioni Lambda.
- Non utilizzare informazioni sensibili nei nomi e negli ID delle risorse Greengrass:
  - Dispositivi principali
  - Componenti
  - Distribuzioni
  - Loggers

Tenere sincronizzato l'orologio del dispositivo

È importante avere un orario preciso sul dispositivo. I certificati X.509 hanno data e ora di scadenza. L'orologio sul dispositivo viene utilizzato per verificare che un certificato server sia ancora valido. Gli orologi dei dispositivi possono andare alla deriva nel tempo o le batterie possono scaricarsi.

Per ulteriori informazioni, consulta la best practice Tenere sincronizzato l'orologio del dispositivo nella Guida per sviluppatori AWS IoT Core.

Raccomandazioni di Cipher Suite

L'impostazione predefinita di Greengrass seleziona le più recenti suite di crittografia TLS disponibili sul dispositivo. Prendi in considerazione la possibilità di disabilitare l'uso delle suite di crittografia precedenti sul dispositivo. Ad esempio, le suite di crittografia CBC.

Per ulteriori informazioni, consulta ilConfigurazione della crittografia Java.

Consulta anche

- Le migliori pratiche di sicurezza inAWS IoT CorenelAWS IoTGuida per gli sviluppatori
- Dieci regole d'oro di sicurezza per le soluzioni IoT industrialisulInternet of Things suAWSBlog ufficiale

Utilizzo AWS IoT Device Tester per AWS IoT Greengrass V2

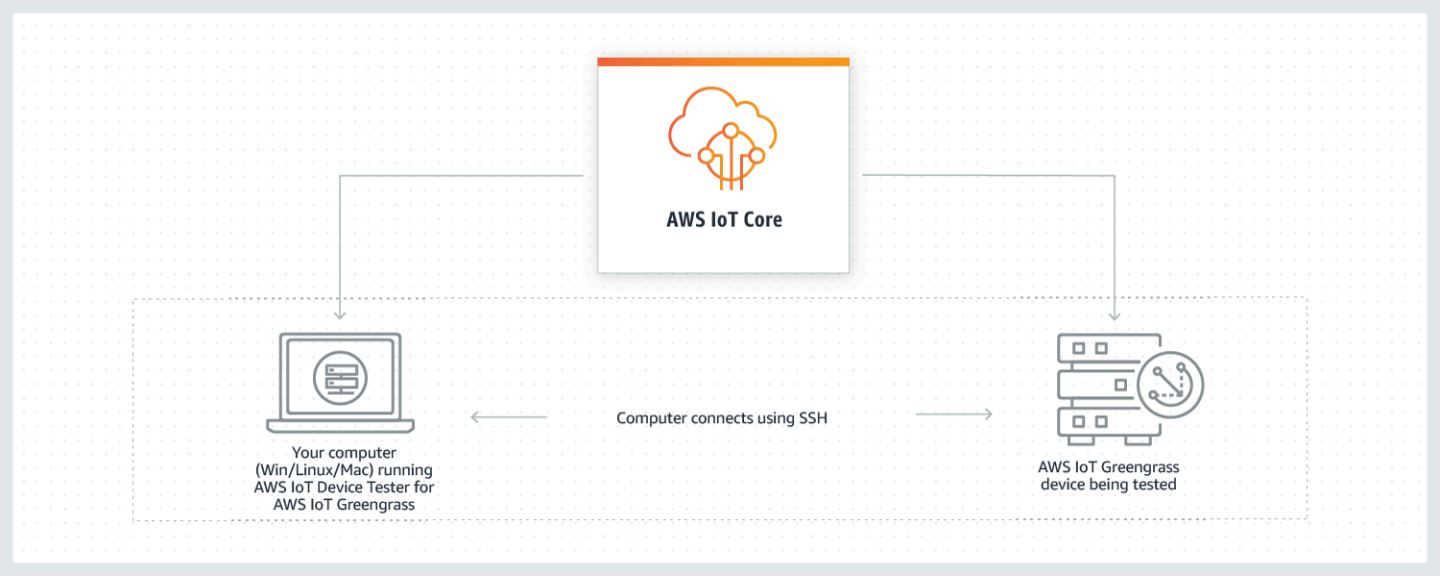
AWS IoTDevice Tester (IDT) è un framework di test scaricabile che consente di convalidare i dispositivi IoT. Puoi utilizzare IDT per eseguire la suite AWS IoT Greengrass di AWS IoT Greengrass qualificazione e creare ed eseguire suite di test personalizzate per i tuoi dispositivi.

IDT per AWS IoT Greengrass viene eseguito sul computer host (Windows, macOS o Linux) connesso al dispositivo da testare. Esegue i test e aggrega i risultati. Inoltre offre un'interfaccia a riga di comando per gestire l'esecuzione di test.

AWS IoT Greengrasssuite di qualificazione

AWS IoT Device TesterUtilizzatelo per la AWS IoT Greengrass versione 2 per verificare che il software AWS IoT Greengrass Core funzioni sull'hardware e sia in grado di comunicare con. Cloud AWS Esegue anche end-to-end test conAWS IoT Core. Ad esempio, verifica che il dispositivo sia in grado di distribuire componenti e aggiornarli.

Se desideri aggiungere il tuo hardware al AWS Partner Device Catalog, esegui la suite di AWS IoT Greengrass qualificazione per generare report di test da inviare. AWS IoT Per ulteriori informazioni, consulta AWS Device Qualification Program.



IDT per AWS IoT Greengrass V2 organizza i test utilizzando i concetti di suite di test e gruppi di test.

- Una suite di test è l'insieme di gruppi di test utilizzati per verificare che un dispositivo funzioni con versioni particolari di AWS IoT Greengrass.

- Un gruppo di test è l'insieme di singoli test relativi a una particolare funzionalità, come la distribuzione dei componenti.

Per ulteriori informazioni, consulta Usa IDT per gestire la suite di AWS IoT Greengrass qualifiche.

Suite di test personalizzate

A partire da IDT v4.0.1, IDT per AWS IoT Greengrass V2 combina una configurazione di configurazione e un formato di risultati standardizzati con un ambiente di suite di test che consente di sviluppare suite di test personalizzate per i dispositivi e il software del dispositivo. Potete aggiungere test personalizzati per la vostra convalida interna o fornirli ai clienti per la verifica dei dispositivi.

Il modo in cui un testwriter configura una suite di test personalizzata determina le configurazioni delle impostazioni necessarie per eseguire suite di test personalizzate. Per ulteriori informazioni, consulta Usa IDT per sviluppare ed eseguire le tue suite di test.

Versioni supportate di AWS IoT Device Tester for AWS IoT Greengrass V2

Questo argomento elenca le versioni supportate di IDT per V2. AWS IoT Greengrass Come best practice, ti consigliamo di utilizzare la versione più recente di IDT per AWS IoT Greengrass V2 che supporti la versione di destinazione di V2. AWS IoT Greengrass Le nuove versioni di AWS IoT Greengrass potrebbero richiedere il download di una nuova versione di IDT per V2. AWS IoT Greengrass Riceverai una notifica quando inizi un'esecuzione di test se IDT per AWS IoT Greengrass V2 non è compatibile con la versione che stai utilizzando. AWS IoT Greengrass

Scaricando il software, l'utente accetta il Contratto di AWS IoT Device Tester licenza.

> **Note**
>
> IDT non supporta l'esecuzione da parte di più utenti da un percorso condiviso, ad esempio una directory NFS o una cartella condivisa di rete Windows. Si consiglia di estrarre il pacchetto IDT in un'unità locale ed eseguire il file binario IDT sulla workstation locale.

Ultima versione IDT per AWS IoT Greengrass V2

È possibile utilizzare questa versione di IDT per AWS IoT Greengrass V2 con la AWS IoT Greengrass versione elencata qui.

IDT v4.9.3 per AWS IoT Greengrass

AWS IoT Greengrass Versioni supportate:

- Greengrass nucleus v2.12.0, v2.11.0, v2.10.0 e v2.9.5

Download del software IDT:

- IDT v4.9.3 con suite di test GGV2Q_2.5.3 per Linux
- IDT v4.9.3 con suite di test GGV2Q_2.5.3 per macOS
- IDT v4.9.3 con suite di test GGV2Q_2.5.3 per Windows

Note di rilascio:

- Risolve un problema nei test dei componenti durante il test di un dispositivo Linux da un host Windows o viceversa.
- Rimuove il `localcomponent` test case dal gruppo `component` di test. Questo test case non è più necessario per la qualificazione.

Note aggiuntive:

- Se il tuo dispositivo utilizza un HSM e stai usando nucleus 2.10.x, esegui la migrazione alla versione 2.11.0 o successiva di Greengrass nucleus.

Versione della suite di test:

`GGV2Q_2.5.3`

- Rilasciato il 2024.04.05

Versioni non supportate di for V2 AWS IoT Device TesterAWS IoT Greengrass

Questo argomento elenca le versioni non supportate di IDT per V2. AWS IoT Greengrass Le versioni non supportate non ricevono correzioni di bug o aggiornamenti. Per ulteriori informazioni, consulta the section called “Politica di supporto AWS IoT Device Tester per AWS IoT Greengrass”.

IDT v4.9.2 per AWS IoT Greengrass

Note di rilascio:

- Risolve un problema a causa del quale la suite di test Lambda fallisce a causa della obsolescenza di Java 8.

Versione della suite di test:

`GGV2Q_2.5.2`

- Rilasciato il 2024.03.18

IDT v4.9.1 per AWS IoT Greengrass

Note di rilascio:

- Consente di convalidare e qualificare i dispositivi che eseguono le versioni del software AWS IoT Greengrass Core 2.12.0, 2.11.0, 2.10.0 e 2.9.5.
- Correzioni di bug minori.

Versione della suite di test:

`GGV2Q_2.5.1`

- Rilasciato il 2023.10.05

IDT v4.7.0 per AWS IoT Greengrass

AWS IoT Greengrass Versioni supportate:

- Greengrass nucleus v2.11.0, v2.10.0 e v2.9.5

Note di rilascio:

- Consente di convalidare e qualificare i dispositivi che eseguono le versioni del software Core 2.11.0, 2.10.0 e 2.9.5. AWS IoT Greengrass
- Aggiunge il supporto per memorizzare i valori dei dati utente IDT in AWS Systems Manager Parameter Store e recuperarli nella configurazione utilizzando la sintassi segnaposto.
- Correzioni di bug minori.

Versione della suite di test:

`GGV2Q_2.5.0`

- Rilasciato il 13 dicembre 2022.

IDT v4.5.11 per AWS IoT Greengrass

Note di rilascio:

- Consente di convalidare e qualificare i dispositivi che eseguono le versioni del software AWS IoT Greengrass Core 2.9.1, 2.9.0, 2.8.1, 2.8.0, 2.7.0 e 2.6.0.

- Aggiunge il supporto per testare PreInstalled Greengrass su un dispositivo principale.
- Correzioni di bug minori.

Versione della suite di test:

`GGV2Q_2.4.1`

- Rilasciato il 2022.10.13

IDT v4.5.8 per AWS IoT Greengrass

Note di rilascio:

- Consente di convalidare e qualificare i dispositivi che eseguono le versioni del software AWS IoT Greengrass Core 2.7.0, 2.6.0 e 2.5.6.
- Consente di eseguire test con PreInstalled Greengrass su un dispositivo principale.
- Correzioni di bug minori.

Versione della suite di test:

`GGV2Q_2.4.0`

- Rilasciato il 2022.08.12

IDT v4.5.3 per AWS IoT Greengrass

Note di rilascio:

- Consente di convalidare e qualificare i dispositivi che eseguono le versioni del software AWS IoT Greengrass Core 2.7.0, 2.6.0, 2.5.6, 2.5.5, 2.5.4 e 2.5.3.
- Aggiorna DockerApplicationManager il test per utilizzare un'immagine docker basata su ECR.
- Correzioni di bug minori.

Versione della suite di test:

`GGV2Q_2.3.1`

- Rilasciato il 15 aprile 2022

IDT v4.5.1 per AWS IoT Greengrass

Note di rilascio:

- Consente di convalidare e qualificare i dispositivi che eseguono il software Core v2.5.3. AWS IoT Greengrass
- Aggiunge il supporto per la convalida e la qualificazione dei dispositivi basati su Linux che utilizzano un modulo di sicurezza hardware (HSM) per archiviare la chiave privata e il certificato utilizzati dal software Core. AWS IoT Greengrass

- Implementa il nuovo orchestrator di test IDT per la configurazione di suite di test personalizzate. Per ulteriori informazioni, consulta Configurazione dell'orchestratore di test IDT.
- Correzioni di bug minori aggiuntive.

Versione della suite di test:

`GGV2Q_2.3.0`

- Rilasciato il 2022.01.11

IDT v4.4.1 per AWS IoT Greengrass

Note di rilascio:

- Consente di convalidare e qualificare i dispositivi che eseguono il software Core v2.5.2. AWS IoT Greengrass
- Aggiunge il supporto per l'utilizzo di un ruolo IAM definito dall'utente come ruolo di scambio di token che il dispositivo sottoposto a test assume per interagire con le risorse. AWS

  È possibile specificare il ruolo IAM nel `userdata.json` file. Se specificate un ruolo personalizzato, IDT utilizza quel ruolo invece di creare il ruolo di scambio di token predefinito durante l'esecuzione del test.
- Correzioni di bug minori aggiuntive.

Versione della suite di test:

`GGV2Q_2.2.1`

- Rilasciato il 2021.12.12

IDT v4.4.0 per AWS IoT Greengrass

Note di rilascio:

- Consente di convalidare e qualificare i dispositivi che eseguono il software Core v2.5.0. AWS IoT Greengrass
- Aggiunge il supporto per la convalida e la qualificazione dei dispositivi che eseguono il software Core su Windows. AWS IoT Greengrass
- Supporta l'uso della convalida a chiave pubblica per le connessioni di dispositivi Secure Shell (SSH).
- Migliora la politica IAM delle autorizzazioni IDT con le migliori pratiche di sicurezza.
- Correzioni di bug minori aggiuntive.

Versione della suite di test:

`GGV2Q_2.1.0`

- Rilasciato il 2021.11.19

IDT v4.2.0 per AWS IoT Greengrass

Note di rilascio:

- Include il supporto per la qualificazione delle seguenti funzionalità sui dispositivi che eseguono il software AWS IoT Greengrass Core v2.2.0 e versioni successive:
  - Docker: verifica che i dispositivi possano scaricare un'immagine del contenitore Docker da Amazon Elastic Container Registry (Amazon ECR).
  - Apprendimento automatico: verifica che i dispositivi possano eseguire inferenze di machine learning (ML) utilizzando i framework Deep Learning Runtime o Lite ML. TensorFlow
  - Stream Manager: verifica che i dispositivi possano scaricare, installare ed eseguire lo stream manager. AWS IoT Greengrass
- Consente di convalidare e qualificare i dispositivi che eseguono il software AWS IoT Greengrass Core v2.4.0, v2.3.0, v2.2.0 e v2.1.0.
- *Raggruppa i log di test per ogni test case in una cartella < > separata all'interno della directory. test-case-id <device-tester-extract-location>*`/results/`*`<execution-id>`*`/logs/`*`<test-group-id>`*
- Correzioni di bug minori aggiuntive.

Versione della suite di test:

`GGV2Q_2.0.1`

- Rilasciato il 2021.08.31

IDT v4.1.0 per AWS IoT Greengrass

Note di rilascio:

- Consente di convalidare e qualificare i dispositivi che eseguono il software AWS IoT Greengrass Core v2.3.0, v2.2.0, v2.1.0 e v2.0.5.
- Migliora la `userdata.json` configurazione eliminando la necessità di specificare le proprietà and. `GreengrassNucleusVersion` `GreengrassCLIVersion`
- Include il supporto per la qualificazione delle funzionalità Lambda e MQTT per il software AWS IoT Greengrass Core v2.1.0 e versioni successive. È ora possibile utilizzare IDT for AWS IoT Greengrass V2 per verificare che il dispositivo principale sia in grado di eseguire

funzioni Lambda e che il dispositivo possa pubblicare e sottoscrivere argomenti MQTT. AWS IoT Core

- Migliora le funzionalità di registrazione.
- Correzioni di bug minori aggiuntive.

Versione della suite di test:

`GGV2Q_1.1.1`

- Rilasciato il 2021.06.18

IDT v4.0.2 per AWS IoT Greengrass

Note di rilascio:

- Consente di convalidare e qualificare i dispositivi che eseguono il software Core v2.1.0. AWS IoT Greengrass
- Aggiunge il supporto per la qualificazione delle funzionalità Lambda e MQTT per il software AWS IoT Greengrass Core v2.1.0 e versioni successive. È ora possibile utilizzare IDT for AWS IoT Greengrass V2 per verificare che il dispositivo principale sia in grado di eseguire funzioni Lambda e che il dispositivo possa pubblicare e sottoscrivere argomenti MQTT. AWS IoT Core
- Migliora le funzionalità di registrazione.
- Correzioni di bug minori aggiuntive.

Versione della suite di test:

`GGV2Q_1.1.1`

- Rilasciato il 2021.05.05

IDT v4.0.1 per AWS IoT Greengrass

Note di rilascio:

- Consente di convalidare e qualificare i dispositivi che eseguono il software versione 2. AWS IoT Greengrass
- Consente di sviluppare ed eseguire suite di test personalizzate utilizzando AWS IoT Device Tester for. AWS IoT Greengrass Per ulteriori informazioni, consulta Usa IDT per sviluppare ed eseguire le tue suite di test.
- Fornisce applicazioni IDT con firma di codice per macOS e Windows. Su macOS, potrebbe essere necessario concedere un'eccezione di sicurezza per IDT. Per ulteriori informazioni, consulta Eccezione di sicurezza su macOS.

Versione della suite di test:

`GGV2Q_1.0.0`

- Rilasciato il 2020.12.22
- La suite di test esegue solo i test necessari per la qualificazione, a meno che non si imposti il corrispondente `value` nell'array su. `features` `yes`

Scarica IDT per V2 AWS IoT Greengrass

Questo argomento descrive le opzioni di download AWS IoT Device Tester per la AWS IoT Greengrass V2. È possibile utilizzare uno dei seguenti collegamenti per il download del software oppure seguire le istruzioni per scaricare IDT a livello di programmazione.

Argomenti

- Scarica IDT manualmente
- Scarica IDT a livello di codice

Scaricando il software, l'utente accetta il Contratto di licenza.AWS IoT Device Tester

> **Note**
>
> IDT non supporta l'esecuzione da parte di più utenti da un percorso condiviso, ad esempio una directory NFS o una cartella condivisa di rete Windows. Si consiglia di estrarre il pacchetto IDT in un'unità locale ed eseguire il file binario IDT sulla workstation locale.

Scarica IDT manualmente

Questo argomento elenca le versioni supportate di IDT per AWS IoT Greengrass V2. Come best practice, ti consigliamo di utilizzare la versione più recente di IDT per AWS IoT Greengrass V2 che supporti la versione di destinazione di V2. AWS IoT Greengrass Le nuove versioni di AWS IoT Greengrass potrebbero richiedere il download di una nuova versione di IDT per V2. AWS IoT Greengrass Riceverai una notifica quando inizi un'esecuzione di test se IDT per AWS IoT Greengrass V2 non è compatibile con la versione che stai utilizzando. AWS IoT Greengrass

IDT v4.9.3 per AWS IoT Greengrass

AWS IoT Greengrass Versioni supportate:

- Greengrass nucleus v2.12.0, v2.11.0, v2.10.0 e v2.9.5

Download del software IDT:

- IDT v4.9.3 con suite di test GGV2Q_2.5.3 per Linux
- IDT v4.9.3 con suite di test GGV2Q_2.5.3 per macOS
- IDT v4.9.3 con suite di test GGV2Q_2.5.3 per Windows

Note di rilascio:

- Risolve un problema nei test dei componenti durante il test di un dispositivo Linux da un host Windows o viceversa.
- Rimuove il `localcomponent` test case dal gruppo `component` di test. Questo test case non è più necessario per la qualificazione.

Note aggiuntive:

- Se il tuo dispositivo utilizza un HSM e stai usando nucleus 2.10.x, esegui la migrazione alla versione 2.11.0 o successiva di Greengrass nucleus.

Versione della suite di test:

`GGV2Q_2.5.3`

- Rilasciato il 2024.04.05

Scarica IDT a livello di codice

IDT fornisce un'operazione API che puoi utilizzare per recuperare un URL da cui scaricare IDT a livello di codice. Puoi anche utilizzare questa operazione API per verificare se disponi della versione più recente di IDT. Questa operazione API ha il seguente endpoint.

```
https://download.devicetester.iotdevicesecosystem.amazonaws.com/latestidt
```

Per richiamare questa operazione API, è necessario disporre dell'autorizzazione per eseguire l'**`iot-device-tester:LatestIdt`**azione. Includi la tua AWS firma e `iot-device-tester` usala come nome del servizio.

Richiesta API

HostOs — Il sistema operativo del computer host. Seleziona una delle opzioni seguenti:

- `mac`
- `linux`
- `windows`

TestSuiteType — Il tipo di suite di test. Scegliete la seguente opzione:

`GGV2`— IDT per V2 AWS IoT Greengrass

ProductVersion

(Opzionale) La versione del nucleo Greengrass. Il servizio restituisce l'ultima versione compatibile di IDT per quella versione del nucleo Greengrass. Se non si specifica questa opzione, il servizio restituisce la versione più recente di IDT.

Risposta API

La risposta dell'API ha il seguente formato. `DownloadURL`Include un file zip.

```
{
    "Success": True or False,
    "Message": Message,
    "LatestBk": {
        "Version": The version of the IDT binary,
        "TestSuiteVersion": The version of the test suite,
        "DownloadURL": The URL to download the IDT Bundle, valid for one hour
    }
 }
```

Esempi

È possibile fare riferimento ai seguenti esempi per scaricare IDT a livello di programmazione. Questi esempi utilizzano credenziali memorizzate nelle variabili di ambiente e. `AWS_ACCESS_KEY_ID` `AWS_SECRET_ACCESS_KEY` Per seguire le migliori pratiche di sicurezza, non memorizzate le credenziali nel codice.

Example Esempio: download utilizzando cURL versione 7.75.0 o successiva (Mac e Linux)

Se hai cURL versione 7.75.0 o successiva, puoi usare il `aws-sigv4` flag per firmare la richiesta API. Questo esempio utilizza jq per analizzare l'URL di download dalla risposta.

> **Warning**
>
> Il `aws-sigv4` flag richiede che i parametri di query della richiesta curl GET siano nell'ordine di o. HostOs/ProductVersion/TestSuiteType HostOs/TestSuiteType Gli ordini non conformi comporteranno un errore nell'ottenere firme non corrispondenti per la stringa canonica dall'API Gateway.
> Se il parametro opzionale ProductVersion è incluso, è necessario utilizzare una versione del prodotto supportata come documentato in Versioni supportate di for V2. AWS IoT Device TesterAWS IoT Greengrass

- Sostituisci *`us-west-2`* con il tuo. Regione AWS Per l'elenco dei codici regionali, consulta Endpoint regionali.
- Sostituisci *`linux`* con il sistema operativo della tua macchina host.
- Sostituisci *`2.5.3`* con la tua versione di AWS IoT Greengrass nucleus.

```
url=$(curl --request GET "https://
download.devicetester.iotdevicesecosystem.amazonaws.com/latestidt?
HostOs=linux&ProductVersion=2.5.3&TestSuiteType=GGV2" \
--user $AWS_ACCESS_KEY_ID:$AWS_SECRET_ACCESS_KEY \
--aws-sigv4 "aws:amz:us-west-2:iot-device-tester" \
| jq -r '.LatestBk["DownloadURL"]')

curl $url --output devicetester.zip
```

Example Esempio: download utilizzando una versione precedente di cURL (Mac e Linux)

È possibile utilizzare il seguente comando cURL con una AWS firma da firmare e calcolare. Per ulteriori informazioni su come firmare e calcolare una AWS firma, consulta Firmare le richieste AWS API.

- Sostituisci *`linux`* con il sistema operativo della tua macchina host.
- Sostituisci *`Timestamp`* con data e ora, ad esempio. **`20220210T004606Z`**

- Sostituisci *Date* con la data, ad esempio. **20220210**
- Sostituisci *AWSRegion*con il tuo Regione AWS. Per l'elenco dei codici regionali, consulta Endpoint regionali.
- Sostituiscilo *AWSSignature*con la AWS firma generata.

```
curl --location --request GET 'https://
download.devicetester.iotdevicesecosystem.amazonaws.com/latestidt?
HostOs=linux&TestSuiteType=GGV2' \
--header 'X-Amz-Date: Timestamp \
--header 'Authorization: AWS4-HMAC-SHA256 Credential=$AWS_ACCESS_KEY_ID/Date/AWSRegion/
iot-device-tester/aws4_request, SignedHeaders=host;x-amz-date, Signature=AWSSignature'
```

Example Esempio: download utilizzando uno script Python

Questo esempio utilizza la libreria di richieste Python. Questo esempio è adattato dall'esempio Python per Sign an AWS API request nel AWS General Reference.

- Sostituisci *us-west-2* con la tua Regione. Per l'elenco dei codici regionali, vedi Endpoint regionali.
- Sostituisci *linux* con il sistema operativo della tua macchina host.

```
# Copyright 2010-2022 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
#License is located at
#
# http://aws.amazon.com/apache2.0/
#
# This file is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS
# OF ANY KIND, either express or implied. See the License for the specific
# language governing permissions and limitations under the License.

# See: http://docs.aws.amazon.com/general/latest/gr/sigv4_signing.html
# This version makes a GET request and passes the signature
# in the Authorization header.
import sys, os, base64, datetime, hashlib, hmac
import requests # pip install requests
# ************* REQUEST VALUES *************
```

```
method = 'GET'
service = 'iot-device-tester'
host = 'download.devicetester.iotdevicesecosystem.amazonaws.com'
region = 'us-west-2'
endpoint = 'https://download.devicetester.iotdevicesecosystem.amazonaws.com/latestidt'
request_parameters = 'HostOs=linux&TestSuiteType=GGV2'

# Key derivation functions. See:
# http://docs.aws.amazon.com/general/latest/gr/signature-v4-examples.html#signature-v4-
examples-python
def sign(key, msg):
    return hmac.new(key, msg.encode('utf-8'), hashlib.sha256).digest()

def getSignatureKey(key, dateStamp, regionName, serviceName):
    kDate = sign(('AWS4' + key).encode('utf-8'), dateStamp)
    kRegion = sign(kDate, regionName)
    kService = sign(kRegion, serviceName)
    kSigning = sign(kService, 'aws4_request')
    return kSigning

# Read AWS access key from env. variables or configuration file. Best practice is NOT
# to embed credentials in code.
access_key = os.environ.get('AWS_ACCESS_KEY_ID')
secret_key = os.environ.get('AWS_SECRET_ACCESS_KEY')
if access_key is None or secret_key is None:
    print('No access key is available.')
    sys.exit()

# Create a date for headers and the credential string
t = datetime.datetime.utcnow()
amzdate = t.strftime('%Y%m%dT%H%M%SZ')
datestamp = t.strftime('%Y%m%d') # Date w/o time, used in credential scope

# ************* TASK 1: CREATE A CANONICAL REQUEST *************
# http://docs.aws.amazon.com/general/latest/gr/sigv4-create-canonical-request.html
# Step 1 is to define the verb (GET, POST, etc.)--already done.
# Step 2: Create canonical URI--the part of the URI from domain to query
# string (use '/' if no path)
canonical_uri = '/latestidt'
# Step 3: Create the canonical query string. In this example (a GET request),
# request parameters are in the query string. Query string values must
# be URL-encoded (space=%20). The parameters must be sorted by name.
# For this example, the query string is pre-formatted in the request_parameters
 variable.
```

```
canonical_querystring = request_parameters
# Step 4: Create the canonical headers and signed headers. Header names
# must be trimmed and lowercase, and sorted in code point order from
# low to high. Note that there is a trailing \n.
canonical_headers = 'host:' + host + '\n' + 'x-amz-date:' + amzdate + '\n'
# Step 5: Create the list of signed headers. This lists the headers
# in the canonical_headers list, delimited with ";" and in alpha order.
# Note: The request can include any headers; canonical_headers and
# signed_headers lists those that you want to be included in the
# hash of the request. "Host" and "x-amz-date" are always required.
signed_headers = 'host;x-amz-date'
# Step 6: Create payload hash (hash of the request body content). For GET
# requests, the payload is an empty string ("").
payload_hash = hashlib.sha256(('').encode('utf-8')).hexdigest()
# Step 7: Combine elements to create canonical request
canonical_request = method + '\n' + canonical_uri + '\n' + canonical_querystring + '\n'
 + canonical_headers + '\n' + signed_headers + '\n' + payload_hash

# ************* TASK 2: CREATE THE STRING TO SIGN*************
# Match the algorithm to the hashing algorithm you use, either SHA-1 or
# SHA-256 (recommended)
algorithm = 'AWS4-HMAC-SHA256'
credential_scope = datestamp + '/' + region + '/' + service + '/' + 'aws4_request'
string_to_sign = algorithm + '\n' +  amzdate + '\n' +  credential_scope + '\n' +
 hashlib.sha256(canonical_request.encode('utf-8')).hexdigest()
# ************* TASK 3: CALCULATE THE SIGNATURE *************
# Create the signing key using the function defined above.
signing_key = getSignatureKey(secret_key, datestamp, region, service)
# Sign the string_to_sign using the signing_key
signature = hmac.new(signing_key, (string_to_sign).encode('utf-8'),
 hashlib.sha256).hexdigest()

# ************* TASK 4: ADD SIGNING INFORMATION TO THE REQUEST *************
# The signing information can be either in a query string value or in
# a header named Authorization. This code shows how to use a header.
# Create authorization header and add to request headers
authorization_header = algorithm + ' ' + 'Credential=' + access_key + '/' +
 credential_scope + ', ' +  'SignedHeaders=' + signed_headers + ', ' + 'Signature=' +
 signature
# The request can include any headers, but MUST include "host", "x-amz-date",
# and (for this scenario) "Authorization". "host" and "x-amz-date" must
# be included in the canonical_headers and signed_headers, as noted
# earlier. Order here is not significant.
```

```
# Python note: The 'host' header is added automatically by the Python 'requests'
 library.
headers = {'x-amz-date':amzdate, 'Authorization':authorization_header}

# ************* SEND THE REQUEST *************
request_url = endpoint + '?' + canonical_querystring
print('\nBEGIN REQUEST++++++++++++++++++++++++++++++++++++')
print('Request URL = ' + request_url)
response = requests.get(request_url, headers=headers)
print('\nRESPONSE++++++++++++++++++++++++++++++++++++')
print('Response code: %d\n' % response.status_code)
print(response.text)

download_url = response.json()["LatestBk"]["DownloadURL"]
r = requests.get(download_url)
open('devicetester.zip', 'wb').write(r.content)
```

Usa IDT per gestire la suite di AWS IoT Greengrass qualifiche

È possibile utilizzare AWS IoT Device Tester for AWS IoT Greengrass V2 per verificare che il software AWS IoT Greengrass Core funzioni sull'hardware e possa comunicare con. Cloud AWS Esegue anche end-to-end test con AWS IoT Core. Ad esempio, verifica che il dispositivo sia in grado di distribuire componenti e aggiornarli.

Oltre ai dispositivi di test, IDT for AWS IoT Greengrass V2 crea risorse (ad esempio AWS IoT oggetti, gruppi e così via) all'interno dell'utente per Account AWS facilitare il processo di qualificazione.

Per creare queste risorse, IDT for AWS IoT Greengrass V2 utilizza AWS le credenziali configurate nel `config.json` file per effettuare chiamate API per conto dell'utente. Il provisioning di queste risorse viene effettuato varie volte nel corso di un test.

Quando utilizzi IDT for AWS IoT Greengrass V2 per eseguire la suite di AWS IoT Greengrass qualifiche, esegue i seguenti passaggi:

1. Carica e convalida la configurazione del dispositivo e delle credenziali.
2. Esegue i test selezionati con le risorse locali e cloud richieste.
3. Esegue la pulizia di risorse locali e cloud.
4. Genera i report di test che indicano se la scheda ha superato i test richiesti per la qualifica.

Versioni della suite di test

IDT per AWS IoT Greengrass V2 organizza i test in suite di test e gruppi di test.

- Una suite di test è l'insieme di gruppi di test utilizzati per verificare che un dispositivo funzioni con versioni particolari di AWS IoT Greengrass.
- Un gruppo di test è l'insieme di singoli test relativi a una particolare funzionalità, come la distribuzione dei componenti.

Le suite di test vengono versionate utilizzando un *major*.*minor*.*patch* formato, ad esempio. `GGV2Q_1.0.0` Quando scarichi IDT, il pacchetto include l'ultima versione della suite di qualificazione Greengrass.

> **Important**
>
> I test delle versioni non supportate della suite di test non sono validi per la qualifica del dispositivo. IDT non stampa i report di qualifica per le versioni non supportate. Per ulteriori informazioni, consulta the section called “Politica di supporto AWS IoT Device Tester per AWS IoT Greengrass”.
>
> È possibile eseguire l'`list-supported-products`operazione per elencare le versioni AWS IoT Greengrass e le suite di test supportate dalla versione corrente di IDT.

Descrizioni dei gruppi di test

Gruppi di test richiesti per la qualificazione principale

Questi gruppi di test sono necessari per qualificare il dispositivo AWS IoT Greengrass V2 per il AWS Partner Device Catalog.

Dipendenze principali

Verifica che il dispositivo soddisfi tutti i requisiti software e hardware per il software AWS IoT Greengrass Core. Questo gruppo di test include il seguente test case:

Versione Java

Verifica che la versione Java richiesta sia installata sul dispositivo in prova. AWS IoT Greengrass richiede Java 8 o versione successiva.

PreTest Convalida

Verifica che il dispositivo soddisfi i requisiti software per eseguire i test.

- Per i dispositivi basati su Linux, questo test verifica se il dispositivo è in grado di eseguire i seguenti comandi Linux:

  `chmod`, `cp`, `echo`, `grep`, `kill`, `ln`, `mkinfo`, `ps`, `rm`, `sh`, `uname`

- Per i dispositivi basati su Windows, questo test verifica se sul dispositivo è installato il seguente software Microsoft:

  Powershell v5.1 o versione successiva, .NET v4.6.1 o versione successiva, Visual C++ 2017 o versione successiva, utilità PsExec

Controllo della versione

Verifica che la versione AWS IoT Greengrass fornita sia compatibile con la versione di AWS IoT Device Tester che stai utilizzando.

Componente

Verifica che il dispositivo sia in grado di distribuire componenti e aggiornarli. Questo gruppo di test include i seguenti test:

Componente cloud

Convalida la funzionalità del dispositivo per i componenti cloud.

Componente locale

Convalida la funzionalità del dispositivo per i componenti locali.

Lambda

Questo test non è applicabile ai dispositivi basati su Windows.

Verifica che il dispositivo possa implementare componenti della funzione Lambda che utilizzano il runtime Java e che le funzioni Lambda possano utilizzare argomenti AWS IoT Core MQTT come fonti di eventi per i messaggi di lavoro.

MQTT

Verifica che il dispositivo sia in grado di sottoscrivere e pubblicare su argomenti MQTT. AWS IoT Core

Gruppi di test facoltativi

> **Note**
>
> Questi gruppi di test sono opzionali e vengono utilizzati solo per dispositivi core Greengrass idonei basati su Linux. Se scegli di qualificarti per i test opzionali, il tuo dispositivo viene elencato con funzionalità aggiuntive nel Catalogo dei dispositivi. AWS Partner

Dipendenze Docker

Verifica che il dispositivo soddisfi tutte le dipendenze tecniche richieste per utilizzare il componente Docker application AWS manager () fornito. `aws.greengrass.DockerApplicationManager`

Qualifica di Docker Application Manager

Verifica che il dispositivo sia in grado di scaricare un'immagine del contenitore Docker da Amazon ECR.

Dipendenze dal Machine Learning

Verifica che il dispositivo soddisfi tutte le dipendenze tecniche richieste per utilizzare i componenti di machine learning (ML) AWS forniti.

Test di inferenza per il Machine Learning

Verifica che il dispositivo sia in grado di eseguire inferenze ML utilizzando i framework Deep Learning Runtime e TensorFlow Lite ML.

Dipendenze di Stream Manager

Verifica che il dispositivo possa scaricare, installare ed eseguire lo AWS IoT Greengrass stream manager.

Integrazione della sicurezza hardware (HSI)

> **Note**
>
> Questo test è disponibile in IDT v4.5.1 e versioni successive solo per i dispositivi basati su Linux. AWS IoT Greengrass attualmente non supporta l'integrazione della sicurezza hardware per i dispositivi Windows.

Verifica che il dispositivo sia in grado di autenticare le connessioni ai AWS IoT Greengrass servizi AWS IoT e utilizzando una chiave privata e un certificato archiviati in un modulo di sicurezza hardware (HSM). Questo test verifica inoltre che il componente del provider PKCS #11 AWS fornito possa interfacciarsi con l'HSM utilizzando una libreria PKCS #11 fornita dal fornitore. Per ulteriori informazioni, consulta Integrazione della sicurezza hardware.

Prerequisiti per l'esecuzione della suite di AWS IoT Greengrass qualifiche

Questa sezione descrive i prerequisiti per l'utilizzo di AWS IoT Device Tester (IDT) per. AWS IoT Greengrass

Scarica la versione più recente di for AWS IoT Device TesterAWS IoT Greengrass

Scaricate l'ultima versione di IDT ed estraete il software in una posizione (*< device-tester-extract-location >*) del file system in cui disponete delle autorizzazioni di lettura/scrittura.

> **Note**
>
> IDT non supporta l'esecuzione da parte di più utenti da un percorso condiviso, ad esempio una directory NFS o una cartella condivisa di rete Windows. Si consiglia di estrarre il pacchetto IDT in un'unità locale ed eseguire il file binario IDT sulla workstation locale.
> In Windows esiste un limite di lunghezza del percorso di 260 caratteri. Se stai usando Windows, estrai IDT in una directory root come `C:\` o `D:\` per mantenere i percorsi entro il limite di 260 caratteri.

Scaricate il software AWS IoT Greengrass

IDT for AWS IoT Greengrass V2 verifica la compatibilità del dispositivo con una versione specifica di. AWS IoT Greengrass Esegui il seguente comando per scaricare il software AWS IoT Greengrass Core in un file denominato. `aws.greengrass.nucleus.zip` Sostituisci *`la versione`* con una versione del componente nucleus supportata per la tua versione IDT.

Linux or Unix

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip >
 aws.greengrass.nucleus.zip
```

Windows Command Prompt (CMD)

```
curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip >
 aws.greengrass.nucleus.zip
```

PowerShell

```
iwr -Uri https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-version.zip -
OutFile aws.greengrass.nucleus.zip
```

Inserite il `aws.greengrass.nucleus.zip` file scaricato nella cartella*<device-tester-extract-location>*`/products/`.

> **Note**
>
> Non posizionare più file in questa directory per lo stesso sistema operativo e architettura.

Crea e configura un Account AWS

Prima di utilizzarlo AWS IoT Device Tester per la AWS IoT Greengrass versione 2, è necessario eseguire le seguenti operazioni:

1. Configurare unAccount AWS. Se ne hai già unoAccount AWS, vai al passaggio 2.
2. Configura le autorizzazioni per IDT.

Queste autorizzazioni dell'account consentono a IDT di accedere ai AWS servizi e creare AWS risorse, come AWS IoT oggetti e AWS IoT Greengrass componenti, per tuo conto.

Per creare queste risorse, IDT for AWS IoT Greengrass V2 utilizza AWS le credenziali configurate nel `config.json` file per effettuare chiamate API per conto dell'utente. Il provisioning di queste risorse viene effettuato varie volte nel corso di un test.

> **Note**
>
> Sebbene la maggior parte dei test sia idonea per il piano AWS gratuito, è necessario fornire una carta di credito quando ci si iscrive a un. Account AWS Per ulteriori informazioni, consulta Perché ho bisogno di un metodo di pagamento se il mio account è coperto dal livello gratuito?

Passo 1: Configurare un Account AWS

In questo passaggio, crea e configura unAccount AWS. Se disponi già di un Account AWS, passa a the section called “Fase 2: configurazione delle autorizzazioni per IDT”.

Se non disponi di un Account AWS, completa la procedura seguente per crearne uno.

Per registrarsi a un Account AWS

1. Apri la pagina all'indirizzo https://portal.aws.amazon.com/billing/signup.
2. Segui le istruzioni online.

   Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

   Durante la registrazione di un Account AWS, viene creato un Utente root dell'account AWS. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWSnell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente amministrativo e utilizza solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

Per creare un utente amministratore, scegli una delle seguenti opzioni.

Scelta di un modo per gestire il tuo amministratore	Per	Come	Puoi anche
In IAM Identity Center (Consigliato)	Usa credenziali a breve termine per accedere a AWS. Ciò è in linea con le best practice per la sicurezza. Per informazioni sulle best practice, consulta	Segui le istruzioni riportate in Nozioni di base nella Guida per l'utente di AWS IAM Identity Center.	Configura l'accesso programmatico seguendo quanto riportato in Configurazione della AWS CLI per utilizzare AWS IAM Identity Center nella Guida per l'utente di AWS Command Line Interface.

Scelta di un modo per gestire il tuo amministratore	Per	Come	Puoi anche
	Best practice per la sicurezza in IAM nella Guida per l'utente di IAM.		
In IAM (Non consigliato)	Usa credenziali a lungo termine per accedere a AWS.	Segui le istruzioni in Creazione del primo utente e gruppo di utenti IAM di amministrazione nella Guida per l'utente di IAM.	Configura l'accesso programmatico seguendo quanto riportato in Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente di IAM.

Fase 2: configurazione delle autorizzazioni per IDT

In questo passaggio, configura le autorizzazioni utilizzate da IDT per AWS IoT Greengrass V2 per eseguire test e raccogliere dati sull'utilizzo di IDT. Puoi utilizzare AWS Management Consoleor AWS Command Line Interface(AWS CLI) per creare una policy IAM e un utente di test per IDT, quindi allegare le policy all'utente. Se hai già creato un utente di prova per IDT, vai a. Configura il tuo dispositivo per eseguire test IDT

Per configurare le autorizzazioni per IDT (Console)

1. Accedere alla console IAM.
2. Creare un criterio gestito dal cliente che concede le autorizzazioni per creare ruoli con autorizzazioni specifiche.
    a. Nel riquadro di navigazione, seleziona Policy e quindi Crea policy.
    b. Se non lo utilizzi PreInstalled, nella scheda JSON, sostituisci il contenuto segnaposto con la seguente politica. Se lo stai utilizzando PreInstalled, procedi al passaggio successivo.

```
{
    "Version":"2012-10-17",
    "Statement":[
    {
      "Sid":"passRoleForResources",
      "Effect":"Allow",
      "Action":"iam:PassRole",
      "Resource":"arn:aws:iam::*:role/idt-*",
      "Condition":{
        "StringEquals":{
          "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid":"lambdaResources",
      "Effect":"Allow",
      "Action":[
        "lambda:CreateFunction",
        "lambda:PublishVersion",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource":[
        "arn:aws:lambda:*:*:function:idt-*"
      ]
    },
    {
      "Sid":"iotResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateThing",
        "iot:DeleteThing",
        "iot:DescribeThing",
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:AddThingToThingGroup",
        "iot:RemoveThingFromThingGroup",
```

```
        "iot:AttachThingPrincipal",
        "iot:DetachThingPrincipal",
        "iot:UpdateCertificate",
        "iot:DeleteCertificate",
        "iot:CreatePolicy",
        "iot:AttachPolicy",
        "iot:DetachPolicy",
        "iot:DeletePolicy",
        "iot:GetPolicy",
        "iot:Publish",
        "iot:TagResource",
        "iot:ListThingPrincipals",
        "iot:ListAttachedPolicies",
        "iot:ListTargetsForPolicy",
        "iot:ListThingGroupsForThing",
        "iot:ListThingsInThingGroup",
        "iot:CreateJob",
        "iot:DescribeJob",
        "iot:DescribeJobExecution",
        "iot:CancelJob"
      ],
      "Resource":[
        "arn:aws:iot:*:*:thing/idt-*",
        "arn:aws:iot:*:*:thinggroup/idt-*",
        "arn:aws:iot:*:*:policy/idt-*",
        "arn:aws:iot:*:*:cert/*",
        "arn:aws:iot:*:*:topic/idt-*",
        "arn:aws:iot:*:*:job/*"
      ]
    },
    {
      "Sid":"s3Resources",
      "Effect":"Allow",
      "Action":[
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:ListBucketVersions",
        "s3:DeleteBucket",
        "s3:PutObjectTagging",
        "s3:PutBucketTagging"
```

```
      ],
      "Resource":"arn:aws:s3::*:idt-*"
    },
    {
      "Sid":"roleAliasResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateRoleAlias",
        "iot:DescribeRoleAlias",
        "iot:DeleteRoleAlias",
        "iot:TagResource",
        "iam:GetRole"
      ],
      "Resource":[
        "arn:aws:iot:*:*:rolealias/idt-*",
        "arn:aws:iam::*:role/idt-*"
      ]
    },
    {
      "Sid":"idtExecuteAndCollectMetrics",
      "Effect":"Allow",
      "Action":[
        "iot-device-tester:SendMetrics",
        "iot-device-tester:SupportedVersion",
        "iot-device-tester:LatestIdt",
        "iot-device-tester:CheckVersion",
        "iot-device-tester:DownloadTestSuite"
      ],
      "Resource":"*"
    },
    {
      "Sid":"genericResources",
      "Effect":"Allow",
      "Action":[
        "greengrass:*",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:ListThings",
        "iot:DescribeEndpoint",
        "iot:CreateKeysAndCertificate"
      ],
      "Resource":"*"
    },
    {
```

```
      "Sid":"iamResourcesUpdate",
      "Effect":"Allow",
      "Action":[
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:TagRole",
        "iam:TagPolicy",
        "iam:GetPolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy"
      ],
      "Resource":[
        "arn:aws:iam::*:role/idt-*",
        "arn:aws:iam::*:policy/idt-*"
      ]
    }
  ]
}
```

c. Se lo stai utilizzando PreInstalled, nella scheda JSON, sostituisci il contenuto segnaposto con la seguente politica. Assicurati di:

- Sostituite *`ThingName`* e *`ThingGroup`* nell'`iotResources`istruzione con il nome e il gruppo di oggetti che sono stati creati durante l'installazione di Greengrass sul dispositivo in fase di test (DUT) per aggiungere le autorizzazioni.
- Sostituisci *`PassRole`* e *`RoleAlias`* nell'istruzione e nell'istruzione con `roleAliasResources` i ruoli creati durante `passRoleForResources` l'installazione di Greengrass sul tuo DUT.

```
{
    "Version":"2012-10-17",
    "Statement":[
    {
      "Sid":"passRoleForResources",
      "Effect":"Allow",
      "Action":"iam:PassRole",
      "Resource":"arn:aws:iam::*:role/passRole",
```

```
      "Condition":{
        "StringEquals":{
          "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid":"lambdaResources",
      "Effect":"Allow",
      "Action":[
        "lambda:CreateFunction",
        "lambda:PublishVersion",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource":[
        "arn:aws:lambda:*:*:function:idt-*"
      ]
    },
    {
      "Sid":"iotResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateThing",
        "iot:DeleteThing",
        "iot:DescribeThing",
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:AddThingToThingGroup",
        "iot:RemoveThingFromThingGroup",
        "iot:AttachThingPrincipal",
        "iot:DetachThingPrincipal",
        "iot:UpdateCertificate",
        "iot:DeleteCertificate",
        "iot:CreatePolicy",
        "iot:AttachPolicy",
        "iot:DetachPolicy",
        "iot:DeletePolicy",
        "iot:GetPolicy",
```

```
        "iot:Publish",
        "iot:TagResource",
        "iot:ListThingPrincipals",
        "iot:ListAttachedPolicies",
        "iot:ListTargetsForPolicy",
        "iot:ListThingGroupsForThing",
        "iot:ListThingsInThingGroup",
        "iot:CreateJob",
        "iot:DescribeJob",
        "iot:DescribeJobExecution",
        "iot:CancelJob"
      ],
      "Resource":[
        "arn:aws:iot:*:*:thing/thingName",
        "arn:aws:iot:*:*:thinggroup/thingGroup",
        "arn:aws:iot:*:*:policy/idt-*",
        "arn:aws:iot:*:*:cert/*",
        "arn:aws:iot:*:*:topic/idt-*",
        "arn:aws:iot:*:*:job/*"
      ]
    },
    {
      "Sid":"s3Resources",
      "Effect":"Allow",
      "Action":[
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:ListBucketVersions",
        "s3:DeleteBucket",
        "s3:PutObjectTagging",
        "s3:PutBucketTagging"
      ],
      "Resource":"arn:aws:s3::*:idt-*"
    },
    {
      "Sid":"roleAliasResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateRoleAlias",
        "iot:DescribeRoleAlias",
```

```
        "iot:DeleteRoleAlias",
        "iot:TagResource",
        "iam:GetRole"
      ],
      "Resource":[
        "arn:aws:iot:*:*:rolealias/roleAlias",
        "arn:aws:iam::*:role/idt-*"
      ]
    },
    {
      "Sid":"idtExecuteAndCollectMetrics",
      "Effect":"Allow",
      "Action":[
        "iot-device-tester:SendMetrics",
        "iot-device-tester:SupportedVersion",
        "iot-device-tester:LatestIdt",
        "iot-device-tester:CheckVersion",
        "iot-device-tester:DownloadTestSuite"
      ],
      "Resource":"*"
    },
    {
      "Sid":"genericResources",
      "Effect":"Allow",
      "Action":[
        "greengrass:*",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:ListThings",
        "iot:DescribeEndpoint",
        "iot:CreateKeysAndCertificate"
      ],
      "Resource":"*"
    },
    {
      "Sid":"iamResourcesUpdate",
      "Effect":"Allow",
      "Action":[
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
```

```
        "iam:TagRole",
        "iam:TagPolicy",
        "iam:GetPolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy"
      ],
      "Resource":[
        "arn:aws:iam::*:role/idt-*",
        "arn:aws:iam::*:policy/idt-*"
      ]
    }
  ]
}
```

> **Note**
>
> Se desideri utilizzare un ruolo IAM personalizzato come ruolo di scambio di token per il dispositivo in esame, assicurati di aggiornare l'`roleAliasResources`istruzione e l'`passRoleForResources`istruzione nella tua policy per consentire la risorsa del tuo ruolo IAM personalizzato.

d. Scegli Esamina la policy.

e. Per Nome, immetti **`IDTGreengrassIAMPermissions`**. In Riepilogo, esaminare le autorizzazioni concesse dai criteri.

f. Scegli Crea policy.

3. Crea un utente IAM e assegna le autorizzazioni richieste da IDT per. AWS IoT Greengrass

   a. Crea un utente IAM. Segui i passaggi da 1 a 5 in Creazione di utenti IAM (console) nella Guida per l'utente IAM.

   b. Allega le autorizzazioni al tuo utente IAM:

      i. Nella pagina Imposta autorizzazioni, seleziona Collega direttamente policy esistenti.

      ii. Cercare il criterio IDTGreenGrassiamPermissions creato nel passaggio precedente. Selezionare la casella di controllo.

   c. Scegliere Successivo: Tag.

   d. Scegliere Next:Review per visualizzare un riepilogo delle tue scelte.

   e. Selezionare Create user (Crea utente).

f. Per visualizzare le chiavi di accesso dell'utente (ID chiave di accesso e chiavi di accesso segrete), scegliere Mostra accanto alla password e alla chiave di accesso. Per salvare le chiavi di accesso, scegliere Scarica .csv e salvare il file in una posizione sicura. Utilizzerai queste informazioni in seguito per configurare il file AWS delle credenziali.

4. Passaggio successivo: configurare il dispositivo fisico.

Per configurare le autorizzazioni per IDT (AWS CLI)

1. Sul computer, installare e configurare l'AWS CLI se non è già installata. Segui la procedura descritta in Installazione di AWS CLI nella Guida per l'AWS Command Line Interfaceutente.

> **Note**
>
> AWS CLIÈ uno strumento open source che puoi utilizzare per interagire con AWS i servizi dalla tua shell a riga di comando.

2. Creare una policy gestita dal cliente che conceda le autorizzazioni per gestire ruoli IDT e AWS IoT Greengrass.

   a. Se non lo utilizzi PreInstalled, apri un editor di testo e salva il seguente contenuto delle policy in un file JSON. Se lo stai utilizzando PreInstalled, procedi al passaggio seguente.

```
{
    "Version":"2012-10-17",
    "Statement":[
    {
      "Sid":"passRoleForResources",
      "Effect":"Allow",
      "Action":"iam:PassRole",
      "Resource":"arn:aws:iam::*:role/idt-*",
      "Condition":{
        "StringEquals":{
          "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
          ]
        }
      }
    },
```

```
    {
      "Sid":"lambdaResources",
      "Effect":"Allow",
      "Action":[
        "lambda:CreateFunction",
        "lambda:PublishVersion",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
      "Resource":[
        "arn:aws:lambda:*:*:function:idt-*"
      ]
    },
    {
      "Sid":"iotResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateThing",
        "iot:DeleteThing",
        "iot:DescribeThing",
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:AddThingToThingGroup",
        "iot:RemoveThingFromThingGroup",
        "iot:AttachThingPrincipal",
        "iot:DetachThingPrincipal",
        "iot:UpdateCertificate",
        "iot:DeleteCertificate",
        "iot:CreatePolicy",
        "iot:AttachPolicy",
        "iot:DetachPolicy",
        "iot:DeletePolicy",
        "iot:GetPolicy",
        "iot:Publish",
        "iot:TagResource",
        "iot:ListThingPrincipals",
        "iot:ListAttachedPolicies",
        "iot:ListTargetsForPolicy",
        "iot:ListThingGroupsForThing",
        "iot:ListThingsInThingGroup",
        "iot:CreateJob",
        "iot:DescribeJob",
        "iot:DescribeJobExecution",
```

```
        "iot:CancelJob"
      ],
      "Resource":[
        "arn:aws:iot:*:*:thing/idt-*",
        "arn:aws:iot:*:*:thinggroup/idt-*",
        "arn:aws:iot:*:*:policy/idt-*",
        "arn:aws:iot:*:*:cert/*",
        "arn:aws:iot:*:*:topic/idt-*",
        "arn:aws:iot:*:*:job/*"
      ]
    },
    {
      "Sid":"s3Resources",
      "Effect":"Allow",
      "Action":[
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:ListBucketVersions",
        "s3:DeleteBucket",
        "s3:PutObjectTagging",
        "s3:PutBucketTagging"
      ],
      "Resource":"arn:aws:s3::*:idt-*"
    },
    {
      "Sid":"roleAliasResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateRoleAlias",
        "iot:DescribeRoleAlias",
        "iot:DeleteRoleAlias",
        "iot:TagResource",
        "iam:GetRole"
      ],
      "Resource":[
        "arn:aws:iot:*:*:rolealias/idt-*",
        "arn:aws:iam::*:role/idt-*"
      ]
    },
    {
```

```
      "Sid":"idtExecuteAndCollectMetrics",
      "Effect":"Allow",
      "Action":[
        "iot-device-tester:SendMetrics",
        "iot-device-tester:SupportedVersion",
        "iot-device-tester:LatestIdt",
        "iot-device-tester:CheckVersion",
        "iot-device-tester:DownloadTestSuite"
      ],
      "Resource":"*"
    },
    {
      "Sid":"genericResources",
      "Effect":"Allow",
      "Action":[
        "greengrass:*",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:ListThings",
        "iot:DescribeEndpoint",
        "iot:CreateKeysAndCertificate"
      ],
      "Resource":"*"
    },
    {
      "Sid":"iamResourcesUpdate",
      "Effect":"Allow",
      "Action":[
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:TagRole",
        "iam:TagPolicy",
        "iam:GetPolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy"
      ],
      "Resource":[
        "arn:aws:iam::*:role/idt-*",
        "arn:aws:iam::*:policy/idt-*"
      ]
```

```
    }
  ]
}
```

b. Se lo stai utilizzando PreInstalled, apri un editor di testo e salva il seguente contenuto delle policy in un file JSON. Assicurati di:

- Sostituite *`ThingName`* e *`ThingGroup`* nell'`iotResources`istruzione creata durante l'installazione di Greengrass sul dispositivo in fase di test (DUT) per aggiungere le autorizzazioni.
- Sostituisci *`PassRole`* e *`RoleAlias`* nell'istruzione e nell'istruzione con `roleAliasResources` i ruoli creati durante `passRoleForResources` l'installazione di Greengrass sul tuo DUT.

```
{
    "Version":"2012-10-17",
    "Statement":[
    {
      "Sid":"passRoleForResources",
      "Effect":"Allow",
      "Action":"iam:PassRole",
      "Resource":"arn:aws:iam::*:role/passRole",
      "Condition":{
        "StringEquals":{
          "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid":"lambdaResources",
      "Effect":"Allow",
      "Action":[
        "lambda:CreateFunction",
        "lambda:PublishVersion",
        "lambda:DeleteFunction",
        "lambda:GetFunction"
      ],
```

```
      "Resource":[
        "arn:aws:lambda:*:*:function:idt-*"
      ]
    },
    {
      "Sid":"iotResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateThing",
        "iot:DeleteThing",
        "iot:DescribeThing",
        "iot:CreateThingGroup",
        "iot:DeleteThingGroup",
        "iot:DescribeThingGroup",
        "iot:AddThingToThingGroup",
        "iot:RemoveThingFromThingGroup",
        "iot:AttachThingPrincipal",
        "iot:DetachThingPrincipal",
        "iot:UpdateCertificate",
        "iot:DeleteCertificate",
        "iot:CreatePolicy",
        "iot:AttachPolicy",
        "iot:DetachPolicy",
        "iot:DeletePolicy",
        "iot:GetPolicy",
        "iot:Publish",
        "iot:TagResource",
        "iot:ListThingPrincipals",
        "iot:ListAttachedPolicies",
        "iot:ListTargetsForPolicy",
        "iot:ListThingGroupsForThing",
        "iot:ListThingsInThingGroup",
        "iot:CreateJob",
        "iot:DescribeJob",
        "iot:DescribeJobExecution",
        "iot:CancelJob"
      ],
      "Resource":[
        "arn:aws:iot:*:*:thing/thingName",
        "arn:aws:iot:*:*:thinggroup/thingGroup",
        "arn:aws:iot:*:*:policy/idt-*",
        "arn:aws:iot:*:*:cert/*",
        "arn:aws:iot:*:*:topic/idt-*",
        "arn:aws:iot:*:*:job/*"
```

```
      ]
    },
    {
      "Sid":"s3Resources",
      "Effect":"Allow",
      "Action":[
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObjectVersion",
        "s3:DeleteObject",
        "s3:CreateBucket",
        "s3:ListBucket",
        "s3:ListBucketVersions",
        "s3:DeleteBucket",
        "s3:PutObjectTagging",
        "s3:PutBucketTagging"
      ],
      "Resource":"arn:aws:s3::*:idt-*"
    },
    {
      "Sid":"roleAliasResources",
      "Effect":"Allow",
      "Action":[
        "iot:CreateRoleAlias",
        "iot:DescribeRoleAlias",
        "iot:DeleteRoleAlias",
        "iot:TagResource",
        "iam:GetRole"
      ],
      "Resource":[
        "arn:aws:iot:*:*:rolealias/roleAlias",
        "arn:aws:iam::*:role/idt-*"
      ]
    },
    {
      "Sid":"idtExecuteAndCollectMetrics",
      "Effect":"Allow",
      "Action":[
        "iot-device-tester:SendMetrics",
        "iot-device-tester:SupportedVersion",
        "iot-device-tester:LatestIdt",
        "iot-device-tester:CheckVersion",
        "iot-device-tester:DownloadTestSuite"
      ],
```

```
      "Resource":"*"
    },
    {
      "Sid":"genericResources",
      "Effect":"Allow",
      "Action":[
        "greengrass:*",
        "iot:GetThingShadow",
        "iot:UpdateThingShadow",
        "iot:ListThings",
        "iot:DescribeEndpoint",
        "iot:CreateKeysAndCertificate"
      ],
      "Resource":"*"
    },
    {
      "Sid":"iamResourcesUpdate",
      "Effect":"Allow",
      "Action":[
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:CreatePolicy",
        "iam:DeletePolicy",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:TagRole",
        "iam:TagPolicy",
        "iam:GetPolicy",
        "iam:ListAttachedRolePolicies",
        "iam:ListEntitiesForPolicy"
      ],
      "Resource":[
        "arn:aws:iam::*:role/idt-*",
        "arn:aws:iam::*:policy/idt-*"
      ]
    }
  ]
}
```

> **Note**
>
> Se desideri utilizzare un ruolo IAM personalizzato come ruolo di scambio di token per il dispositivo in esame, assicurati di aggiornare

l'`roleAliasResources`istruzione e l'`passRoleForResources`istruzione nella tua policy per consentire la risorsa del tuo ruolo IAM personalizzato.

c. Esegui il comando seguente per creare una policy gestita dal cliente denominata`IDTGreengrassIAMPermissions`. Sostituisci *`policy.json`* con il percorso completo del file JSON creato nel passaggio precedente.

```
aws iam create-policy --policy-name IDTGreengrassIAMPermissions --policy-
document file://policy.json
```

3. Crea un utente IAM e allega le autorizzazioni richieste da IDT for. AWS IoT Greengrass

a. Crea un utente IAM. In questa configurazione di esempio, l'utente viene chiamato `IDTGreengrassUser`.

```
aws iam create-user --user-name IDTGreengrassUser
```

b. Allega la `IDTGreengrassIAMPermissions` policy che hai creato nel passaggio 2 al tuo utente IAM. Sostituisci <account-id>nel comando con l'ID del tuoAccount AWS.

```
aws iam attach-user-policy --user-name IDTGreengrassUser --policy-arn
 arn:aws:iam::<account-id>:policy/IDTGreengrassIAMPermissions
```

4. Creare una chiave di accesso segreta per l'utente.

```
aws iam create-access-key --user-name IDTGreengrassUser
```

Memorizzare l'output in una posizione sicura. Queste informazioni verranno utilizzate successivamente per configurare il file AWS delle credenziali.

5. Passaggio successivo: configurare il dispositivo fisico.

Autorizzazioni di AWS IoT Device Tester

Le seguenti politiche descrivono le AWS IoT Device Tester autorizzazioni.

AWS IoT Device Testerrichiede queste autorizzazioni per il controllo della versione e le funzionalità di aggiornamento automatico.

- `iot-device-tester:SupportedVersion`

Concede AWS IoT Device Tester l'autorizzazione a recuperare l'elenco dei prodotti supportati, delle suite di test e delle versioni IDT.

- `iot-device-tester:LatestIdt`

  Concede AWS IoT Device Tester l'autorizzazione a recuperare l'ultima versione IDT disponibile per il download.

- `iot-device-tester:CheckVersion`

  Concede AWS IoT Device Tester l'autorizzazione a verificare la compatibilità delle versioni per IDT, suite di test e prodotti.

- `iot-device-tester:DownloadTestSuite`

  Concede AWS IoT Device Tester l'autorizzazione a scaricare gli aggiornamenti delle suite di test.

AWS IoT Device Testerutilizza anche la seguente autorizzazione per la segnalazione facoltativa delle metriche:

- `iot-device-tester:SendMetrics`

  Concede l'autorizzazione AWS a raccogliere metriche sull'AWS IoT Device Testeruso interno. Se questa autorizzazione viene omessa, queste metriche non verranno raccolte.

Configura il tuo dispositivo per eseguire test IDT

Per consentire a IDT di eseguire test per la qualificazione dei dispositivi, è necessario configurare il computer host per accedere al dispositivo e configurare le autorizzazioni utente sul dispositivo.

Installa Java sul computer host

A partire da IDT v4.2.0, i test di qualificazione opzionali AWS IoT Greengrass richiedono l'esecuzione di Java.

È possibile utilizzare Java versione 8 o successiva. Ti consigliamo di utilizzare le versioni di supporto a lungo termine di Amazon Corretto o OpenJDK. È richiesta la versione 8 o successiva.

Configurazione del computer host per l'accesso al dispositivo sottoposto a test

IDT viene eseguito sul computer host e deve essere in grado di utilizzare SSH per connettersi al dispositivo. Sono disponibili due opzioni per consentire a IDT di ottenere l'accesso SSH ai dispositivi sottoposti a test:

1. Segui le istruzioni contenute in questa pagina per creare una coppia di chiavi SSH e autorizzare la chiave ad accedere al dispositivo sottoposto a test senza specificare una password.
2. Fornisci un nome utente e una password per ogni dispositivo nel file `device.json`. Per ulteriori informazioni, consulta Configura dispositivo.json.

Puoi utilizzare qualsiasi implementazione SSL per creare una chiave SSH. Le seguenti istruzioni mostrano come utilizzare SSH-KEYGEN o PuTTYgen (per Windows). Se stai utilizzando un'altra implementazione SSL, consulta la documentazione dell'applicazione.

IDT utilizza chiavi SSH per eseguire l'autenticazione con il dispositivo sottoposto a test.

Per creare una chiave SSH con SSH-KEYGEN

1. Crea una chiave SSH.

   Puoi utilizzare il comando Open SSH ssh-keygen per creare una coppia di chiavi SSH. Se disponi già di una coppia di chiavi SSH sul computer host, una best practice è creare una coppia di chiavi SSH appositamente per IDT. In questo modo, dopo aver completato il test, il computer host non può più connettersi al dispositivo senza immettere una password. Ciò consente inoltre di limitare l'accesso al dispositivo remoto solo a coloro che ne hanno bisogno.

   > **Note**
   >
   > Windows non dispone di un client SSH installato. Per informazioni sull'installazione di un client SSH in Windows, consulta Download del software client SSH.

   Il comando ssh-keygen richiede di specificare un nome e un percorso di archiviazione della coppia di chiavi. Per impostazione predefinita, i file della coppia di chiavi sono `id_rsa` (chiave privata) e `id_rsa.pub` (chiave pubblica). In macOS e Linux, il percorso predefinito di questi file è `~/.ssh/`. In Windows, la posizione predefinita è `C:\Users\<user-name>\.ssh`.

Quando richiesto, immetti una frase chiave per proteggere la chiave SSH. Per ulteriori informazioni, consulta l'argomento relativo alla generazione di una nuova chiave SSH.

2. Aggiungi chiavi SSH autorizzate al dispositivo sottoposto a test.

   IDT deve utilizzare la chiave privata SSH per accedere al dispositivo sottoposto al test. Per autorizzare le chiavi SSH private per accedere al dispositivo sottoposto a test, utilizza il comando ssh-copy-id dal computer host. Questo comando aggiunge la chiave pubblica al file `~/.ssh/authorized_keys` nel dispositivo sottoposto a test. Per esempio:

   $ ssh-copy-id ***<remote-ssh-user>***@***<remote-device-ip>***

   *`remote-ssh-user`*Dov'è il nome utente utilizzato per accedere al dispositivo sottoposto a test e *`remote-device-ip`*l'indirizzo IP del dispositivo sottoposto a test su cui eseguire i test. Per esempio:

   ssh-copy-id pi@192.168.1.5

   Quando richiesto, immetti la password per il nome utente specificato nel comando ssh-copy-id.

   ssh-copy-id presuppone che la chiave pubblica sia denominata `id_rsa.pub` e sia archiviata nella posizione predefinita (in macOS e Linux `~/.ssh/` e in Windows `C:\Users\`*`<user-name>`*`\.ssh`). Se hai attribuito un nome diverso alla chiave pubblica o la hai archiviata in una posizione diversa, devi specificare il percorso completo della chiave pubblica SSH con l'opzione -i di ssh-copy-id, ad esempio ssh-copy-id -i ~/my/path/myKey.pub. Per ulteriori informazioni sulla creazione di chiavi SSH e sulla copia di chiavi pubbliche, consulta SSH-COPY-ID.

Per creare una chiave SSH utilizzando PuTTYgen (solo Windows)

1. Assicurati di aver installato il server e il client OpenSSH sul dispositivo sottoposto a test. Per ulteriori informazioni, consulta OpenSSH.
2. Installa PuTTYgen sul dispositivo sottoposto a test.
3. Apri PuTTYgen.
4. Scegli Generate (Genera) e sposta il cursore del mouse all'interno della casella per generare una chiave privata.
5. Dal menu Conversions (Conversioni), scegli Export OpenSSH key (Esporta chiave OpenSSH) e salvare la chiave privata con un'estensione di file `.pem`.

6. Aggiungi la chiave pubblica al file `/home/<user>/.ssh/authorized_keys` sul dispositivo sottoposto a test.

   a. Copia il testo della chiave pubblica dalla finestra PuTTYgen.

   b. Utilizza PuTTY per creare una sessione sul dispositivo sottoposto a test.

      i. Da un prompt dei comandi o dalla finestra Windows Powershell, esegui il comando seguente:

         C:/***<path-to-putty>***/putty.exe -ssh ***<user>***@***<dut-ip-address>***

      ii. Quando richiesto, immetti la password del dispositivo.

      iii. Utilizza vi o un altro editor di testo per aggiungere la chiave pubblica al file `/home/<user>/.ssh/authorized_keys` sul dispositivo sottoposto a test.

7. Aggiorna il file `device.json` con il nome utente, l'indirizzo IP e il percorso al file della chiave privata appena salvato sul computer host per ogni dispositivo sottoposto a test. Per ulteriori informazioni, consulta the section called “Configura dispositivo.json”. Assicurati di fornire il percorso completo e il nome di file per la chiave privata e utilizza le barre ("/"). Ad esempio, per il percorso di Windows `C:\DT\privatekey.pem`, utilizza `C:/DT/privatekey.pem` nel file `device.json`.

Configura le credenziali utente per i dispositivi Windows

Per qualificare un dispositivo basato su Windows, è necessario configurare le credenziali utente nell'LocalSystem account sul dispositivo in esame per i seguenti utenti:

- L'utente Greengrass predefinito ()`ggc_user`.
- L'utente che usi per connetterti al dispositivo in prova. Questo utente viene configurato nel `device.json`file.

È necessario creare ogni utente nell' LocalSystem account sul dispositivo in esame, quindi memorizzare il nome utente e la password dell'utente nell'istanza di Credential Manager dell'LocalSystem account.

Per configurare gli utenti sui dispositivi Windows

1. Apri il prompt dei comandi di Windows (`cmd.exe`) come amministratore.

2. Crea gli utenti nell' LocalSystem account sul dispositivo Windows. Esegui il comando seguente per ogni utente che desideri creare. *Per l'utente Greengrass predefinito, sostituisci il nome utente con.* `ggc_user` Sostituisci *la password* con una password sicura.

   ```
   net user /add user-name password
   ```

3. Scarica e installa l'PsExecutilità di Microsoft sul dispositivo.

4. Utilizzate l' PsExec utilità per memorizzare il nome utente e la password per l'utente predefinito nell'istanza di Credential Manager per l' LocalSystem account.

   Eseguite il comando seguente per ogni utente che desiderate configurare in Credential Manager. *Per l'utente Greengrass predefinito, sostituisci il nome utente con.* `ggc_user` Sostituisci *la password* con la password dell'utente che hai impostato in precedenza.

   ```
   psexec -s cmd /c cmdkey /generic:user-name /user:user-name /pass:password
   ```

   Se si PsExec License Agreementapre, scegli Acceptdi accettare la licenza ed esegui il comando.

   > **Note**
   >
   > Sui dispositivi Windows, l' LocalSystem account esegue il nucleo Greengrass ed è necessario utilizzare l' PsExec utilità per memorizzare le informazioni utente nell'account. LocalSystem L'utilizzo dell'applicazione Credential Manager archivia queste informazioni nell'account Windows dell'utente attualmente connesso, anziché nell'account. LocalSystem

Configurazione delle autorizzazioni utente sul dispositivo

IDT esegue operazioni su varie directory e file in un dispositivo sottoposto a test. Alcune di queste operazioni richiedono autorizzazioni elevate (utilizzando sudo). Per automatizzare queste operazioni, IDT per AWS IoT Greengrass V2 deve essere in grado di eseguire comandi con sudo senza che venga richiesta una password.

Segui questi passaggi sul dispositivo sottoposto a test per consentire al comando sudo di accedere senza che venga richiesta una password.

> **Note**
>
> `username` fa riferimento all'utente SSH utilizzato da IDT per accedere al dispositivo sottoposto a test.

Per aggiungere l'utente al gruppo sudo

1. Sul dispositivo sottoposto a test, esegui `sudo usermod -aG sudo <username>`.
2. Per rendere effettive le modifiche, esci ed esegui di nuovo l'accesso.
3. Per verificare che il nome utente sia stato aggiunto correttamente, esegui sudo echo test. Se non viene è richiesta una password, l'utente è configurato correttamente.
4. Apri il file `/etc/sudoers`, quindi aggiungi la riga seguente alla fine del file:

   ```
   <ssh-username> ALL=(ALL) NOPASSWD: ALL
   ```

Configura un ruolo di scambio di token personalizzato

Puoi scegliere di utilizzare un ruolo IAM personalizzato come ruolo di scambio di token che il dispositivo sottoposto a test assume per interagire con AWS le risorse. Per informazioni sulla creazione di un ruolo IAM, consulta Creating IAM roles nella IAM User Guide.

È necessario soddisfare i seguenti requisiti per consentire a IDT di utilizzare il ruolo IAM personalizzato. Ti consigliamo vivamente di aggiungere solo le azioni politiche minime richieste a questo ruolo.

- Il file di configurazione userdata.json deve essere aggiornato per impostare il parametro su. `GreengrassV2TokenExchangeRole` `true`
- Il ruolo IAM personalizzato deve essere configurato con la seguente politica di fiducia minima:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":[
               "credentials.iot.amazonaws.com",
               "lambda.amazonaws.com",
```

```
                "sagemaker.amazonaws.com"
            ]
         },
         "Action":"sts:AssumeRole"
      }
   ]
}
```

- Il ruolo IAM personalizzato deve essere configurato con la seguente politica di autorizzazioni minime:

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "iot:DescribeCertificate",
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:DescribeLogStreams",
            "iot:Connect",
            "iot:Publish",
            "iot:Subscribe",
            "iot:Receive",
            "iot:ListThingPrincipals",
            "iot:GetThingShadow",
            "iot:UpdateThingShadow",
            "s3:GetBucketLocation",
            "s3:GetObject",
            "s3:PutObject",
            "s3:AbortMultipartUpload",
            "s3:ListMultipartUploadParts"
         ],
         "Resource":"*"
      }
   ]
}
```

- Il nome del ruolo IAM personalizzato deve corrispondere alla risorsa del ruolo IAM specificata nelle autorizzazioni IAM per l'utente di test. Per impostazione predefinita, la policy dell'utente di test consente l'accesso ai ruoli IAM che hanno il `idt-` prefisso nei nomi dei ruoli. Se il nome del tuo

ruolo IAM non utilizza questo prefisso, aggiungi la `arn:aws:iam::*:role/`*`custom-iam-role-name`* risorsa all'`roleAliasResources`istruzione e all'`passRoleForResources`istruzione nella tua policy per l'utente di test, come mostrato negli esempi seguenti:

Example Dichiarazione **`passRoleForResources`**

```
{
   "Sid":"passRoleForResources",
   "Effect":"Allow",
   "Action":"iam:PassRole",
   "Resource":"arn:aws:iam::*:role/custom-iam-role-name",
   "Condition":{
      "StringEquals":{
         "iam:PassedToService":[
            "iot.amazonaws.com",
            "lambda.amazonaws.com",
            "greengrass.amazonaws.com"
         ]
      }
   }
}
```

Example Dichiarazione **`roleAliasResources`**

```
{
   "Sid":"roleAliasResources",
   "Effect":"Allow",
   "Action":[
      "iot:CreateRoleAlias",
      "iot:DescribeRoleAlias",
      "iot:DeleteRoleAlias",
      "iot:TagResource",
      "iam:GetRole"
   ],
   "Resource":[
      "arn:aws:iot:*:*:rolealias/idt-*",
      "arn:aws:iam::*:role/custom-iam-role-name"
   ]
}
```

Configurazione del dispositivo per testare le funzionalità opzionali

Questa sezione descrive i requisiti del dispositivo per eseguire i test IDT per le funzionalità opzionali di Docker e machine learning (ML). Devi assicurarti che il tuo dispositivo soddisfi questi requisiti solo se desideri testare queste funzionalità. Altrimenti, passare a the section called “Configurare le impostazioni IDT”.

Argomenti

Requisiti di qualificazione Docker

IDT per AWS IoT Greengrass V2 fornisce test di qualificazione Docker per verificare che i dispositivi possano utilizzare il componente Docker Application Manager AWS fornito per scaricare immagini di container Docker che è possibile eseguire utilizzando componenti Docker container personalizzati. Per informazioni sulla creazione di componenti Docker personalizzati, consulta. Esegui un contenitore Docker

Per eseguire i test di qualificazione Docker, i dispositivi sottoposti a test devono soddisfare i seguenti requisiti per distribuire il componente Docker Application Manager.

- Docker Engine 1.9.1 o versione successiva installato sul dispositivo principale Greengrass. La versione 20.10 è l'ultima versione verificata per funzionare con il software Core. AWS IoT Greengrass È necessario installare Docker direttamente sul dispositivo principale prima di distribuire componenti che eseguono contenitori Docker.
- Il daemon Docker è stato avviato e funzionante sul dispositivo principale prima di distribuire questo componente.
- L'utente di sistema che esegue un componente del contenitore Docker deve disporre delle autorizzazioni di root o di amministratore oppure è necessario configurare Docker per eseguirlo come utente non root o non amministratore.
  - Sui dispositivi Linux, puoi aggiungere un utente al gruppo senza il quale chiamare i comandi. `docker` `docker` `sudo`
  - Nei dispositivi Windows, è possibile aggiungere un utente al `docker-users` gruppo per richiamare `docker` comandi senza privilegi di amministratore.

Linux or Unix

Per aggiungere `ggc_user` al `docker` gruppo l'utente non root che usi per eseguire i componenti del contenitore Docker, esegui il comando seguente.

```
sudo usermod -aG docker ggc_user
```

Per ulteriori informazioni, consulta Gestire Docker come utente non root.

Windows Command Prompt (CMD)

Per aggiungere al `docker-users` gruppo`ggc_user`, o l'utente che usi per eseguire i componenti del contenitore Docker, esegui il comando seguente come amministratore.

```
net localgroup docker-users ggc_user /add
```

Windows PowerShell

Per aggiungere al `docker-users` gruppo`ggc_user`, o l'utente che usi per eseguire i componenti del contenitore Docker, esegui il comando seguente come amministratore.

```
Add-LocalGroupMember -Group docker-users -Member ggc_user
```

Requisiti di qualificazione ML

IDT for AWS IoT Greengrass V2 fornisce test di qualificazione ML per verificare che i dispositivi possano utilizzare i componenti di machine learning AWS forniti per eseguire l'inferenza ML localmente utilizzando i framework Deep Learning Runtime o Lite ML. TensorFlow Per ulteriori informazioni sull'esecuzione dell'inferenza ML sui dispositivi Greengrass, vedere. Esecuzione dell'inferenza di Machine Learning

Per eseguire i test di qualificazione ML, i dispositivi sottoposti a test devono soddisfare i seguenti requisiti per implementare i componenti di machine learning.

- Sui dispositivi core Greengrass che eseguono Amazon Linux 2 o Ubuntu 18.04, sul dispositivo è installata la versione 2.27 o successiva della GNU C Library (glibc).
- Sui dispositivi ARMv7L, come Raspberry Pi, le dipendenze per OpenCV-Python sono installate sul dispositivo. Esegui il comando seguente per installare le dipendenze.

```
sudo apt-get install libopenjp2-7 libilmbase23 libopenexr-dev libavcodec-dev
 libavformat-dev libswscale-dev libv4l-dev libgtk-3-0 libwebp-dev
```

- I dispositivi Raspberry Pi che eseguono il sistema operativo Raspberry Pi Bullseye devono soddisfare i seguenti requisiti:
  - NumPy 1.22.4 o versione successiva installata sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include una versione precedente di NumPy, quindi è possibile eseguire il seguente comando per l'aggiornamento del dispositivo. NumPy

    ```
    pip3 install --upgrade numpy
    ```

  - Lo stack di fotocamere legacy è abilitato sul dispositivo. Il sistema operativo Raspberry Pi Bullseye include un nuovo stack di fotocamere abilitato di default e non compatibile, quindi è necessario abilitare lo stack di fotocamere precedente.

    Per abilitare lo stack di telecamere precedente

    1. Esegui il seguente comando per aprire lo strumento di configurazione Raspberry Pi.

       ```
       sudo raspi-config
       ```

    2. Seleziona Opzioni di interfaccia.
    3. Seleziona Legacy camera per abilitare lo stack di telecamere legacy.
    4. Riavvia il dispositivo Raspberry Pi.

Requisiti di qualificazione HSM

AWS IoT Greengrassfornisce il componente provider PKCS #11 da integrare con l'Hardware Security Module (HSM) PKCS sul dispositivo. La configurazione HSM dipende dal dispositivo e dal modulo HSM scelto. Se viene fornita la configurazione HSM prevista, documentata nelle impostazioni di configurazione IDT, IDT disporrà delle informazioni necessarie per eseguire questo test di qualificazione delle funzionalità opzionali.

Configurare le impostazioni IDT per eseguire la suite di AWS IoT Greengrass qualifiche

Prima di eseguire i test, è necessario configurare le impostazioni per le AWS credenziali e i dispositivi sul computer host.

Configura le AWS credenziali in config.json

È necessario configurare le credenziali utente IAM nel file. *<device_tester_extract_location>*`/configs/config.json` Utilizza le credenziali per l'utente IDT for AWS IoT Greengrass V2 creato in. the section called “Crea e configura un Account AWS” Puoi specificare le credenziali in uno dei due modi seguenti:

- In un file di credenziali
- Come variabili di ambiente

Configura AWS le credenziali con un file di credenziali

IDT usa lo stesso file delle credenziali di AWS CLI. Per ulteriori informazioni, consulta l'argomento relativo ai file di configurazione e delle credenziali.

La posizione del file delle credenziali varia in base al sistema operativo in uso:

- macOS, Linux: `~/.aws/credentials`
- Windows: `C:\Users\`*`UserName`*`\.aws\credentials`

Aggiungi AWS le tue credenziali al `credentials` file nel seguente formato:

```
[default]
aws_access_key_id = <your_access_key_id>
aws_secret_access_key = <your_secret_access_key>
```

Per configurare IDT per AWS IoT Greengrass V2 in modo che utilizzi le AWS credenziali `credentials` del tuo file, modifica il file come segue: `config.json`

```
{
  "awsRegion": "region",
  "auth": {
    "method": "file",
    "credentials": {
      "profile": "default"
    }
  }
}
```

> **Note**
>
> Se non utilizzate il `default` AWS profilo, assicuratevi di cambiarne il nome nel file. `config.json` Per ulteriori informazioni, consulta l'articolo relativo ai profili denominati.

Configura AWS le credenziali con variabili di ambiente

Le variabili di ambiente sono variabili gestite dal sistema operativo e utilizzate dai comandi di sistema. Non vengono salvate se chiudi la sessione SSH. IDT per AWS IoT Greengrass V2 può utilizzare le variabili di `AWS_SECRET_ACCESS_KEY` ambiente `AWS_ACCESS_KEY_ID` e per memorizzare le credenziali. AWS

Per impostare queste variabili su Linux, macOS o Unix, utilizza export:

```
export AWS_ACCESS_KEY_ID=<your_access_key_id>
export AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
```

Per impostare queste variabili su Windows, utilizza set:

```
set AWS_ACCESS_KEY_ID=<your_access_key_id>
set AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
```

Per configurare IDT per utilizzare le variabili di ambiente, modifica la sezione `auth` nel file `config.json`. Ecco un esempio:

```
{
  "awsRegion": "region",
  "auth": {
    "method": "environment"
  }
}
```

Configura dispositivo.json

Oltre alle AWS credenziali, IDT for AWS IoT Greengrass V2 necessita di informazioni sui dispositivi su cui vengono eseguiti i test. Le informazioni di esempio potrebbero essere l'indirizzo IP, le informazioni di accesso, il sistema operativo e l'architettura della CPU.

Devi fornire queste informazioni utilizzando il modello `device.json` situato in *<device_tester_extract_location>*`/configs/device.json`:

```
[
  {
    "id": "<pool-id>",
    "sku": "<sku>",
    "features": [
      {
        "name": "arch",
        "value": "x86_64 | armv6l | armv7l | aarch64"
      },
      {
        "name": "ml",
        "value": "dlr | tensorflowlite | dlr,tensorflowlite | no"
      },
      {
        "name": "docker",
        "value": "yes | no"
      },
      {
        "name": "streamManagement",
        "value": "yes | no"
      },
      {
        "name": "hsi",
        "value": "hsm | no"
      }
    ],
    "devices": [
      {
        "id": "<device-id>",
        "operatingSystem": "Linux | Windows",
        "connectivity": {
          "protocol": "ssh",
          "ip": "<ip-address>",
          "port": 22,
          "publicKeyPath": "<public-key-path>",
          "auth": {
            "method": "pki | password",
            "credentials": {
              "user": "<user-name>",
              "privKeyPath": "/path/to/private/key",
```

```
            "password": "<password>"
          }
        }
      }
    }
  ]
}
]
```

> **Note**
>
> Specificare `privKeyPath` solo se `method` è impostato su `pki`.
> Specificare `password` solo se `method` è impostato su `password`.

Tutte le proprietà che contengono valori sono obbligatorie, come descritto di seguito:

`id`

Un ID alfanumerico definito dall'utente che identifica in modo univoco una raccolta di dispositivi denominata un pool di dispositivi. I dispositivi che appartengono a un pool devono avere lo stesso hardware. Durante l'esecuzione di una suite di test, i dispositivi del pool vengono utilizzati per parallelizzare il carico di lavoro. Più dispositivi vengono utilizzati per eseguire diversi test.

`sku`

Un valore alfanumerico che identifica in modo univoco il dispositivo sottoposto a test. Il codice SKU viene utilizzato per tenere traccia delle schede qualificate.

> **Note**
>
> Se desideri inserire il tuo dispositivo nel Catalogo dispositivi, lo AWS Partner SKU che specifichi qui deve corrispondere allo SKU che utilizzi nel processo di pubblicazione delle offerte.

`features`

Un array contenente le caratteristiche supportate del dispositivo. Tutte le funzionalità sono obbligatorie.

`arch`

Le architetture del sistema operativo supportate convalidate dall'esecuzione del test. I valori validi sono:

- `x86_64`
- `armv6l`
- `armv7l`
- `aarch64`

`ml`

Verifica che il dispositivo soddisfi tutte le dipendenze tecniche richieste per utilizzare i componenti di machine learning (AWSML) forniti.

L'attivazione di questa funzionalità consente inoltre di verificare che il dispositivo sia in grado di eseguire inferenze ML utilizzando i framework Deep Learning Runtime e TensorFlow Lite ML.

I valori validi sono qualsiasi combinazione di `dlr` and, o. `tensorflowlite` no

`docker`

Verifica che il dispositivo soddisfi tutte le dipendenze tecniche richieste per utilizzare il componente Docker application manager () AWS fornito. `aws.greengrass.DockerApplicationManager`

L'attivazione di questa funzionalità consente inoltre di verificare che il dispositivo possa scaricare un'immagine del contenitore Docker da Amazon ECR.

I valori validi sono qualsiasi combinazione di o. `yes` no

`streamManagement`

Verifica che il dispositivo possa scaricare, installare ed eseguire lo AWS IoT Greengrassstream manager.

I valori validi sono qualsiasi combinazione di `yes` ono.

`hsi`

Verifica che il dispositivo sia in grado di autenticare le connessioni ai AWS IoT Greengrass servizi AWS IoT e utilizzando una chiave privata e un certificato archiviati in un modulo di

sicurezza hardware (HSM). Questo test verifica inoltre che il componente del provider PKCS #11 AWS fornito possa interfacciarsi con l'HSM utilizzando una libreria PKCS #11 fornita dal fornitore. Per ulteriori informazioni, consulta Integrazione della sicurezza hardware.

I valori validi sono `hsm` e `no`.

> **Note**
>
> IDT v4.2.0 e versioni successive supportano il test di, and. `ml` `docker` `streamManagement` Se non desideri testare queste funzionalità, imposta il valore corrispondente su. `no`

> **Note**
>
> Il test di `hsi` è disponibile solo con IDT v4.5.1 e versioni successive.

`devices.id`

Un identificativo univoco definito dall'utente del dispositivo sottoposto a test.

`devices.operatingSystem`

Il sistema operativo del dispositivo. I valori supportati sono `Linux` e `Windows`.

`connectivity.protocol`

Il protocollo di comunicazione utilizzato per comunicare con questo dispositivo. Attualmente, l'unico valore supportato è `ssh` per i dispositivi fisici.

`connectivity.ip`

L'indirizzo IP del dispositivo sottoposto a test.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.port`

Facoltativo. Il numero di porta da utilizzare per le connessioni SSH.

Il valore predefinito è 22.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.publicKeyPath`

Facoltativo. Il percorso completo della chiave pubblica utilizzata per autenticare le connessioni al dispositivo in esame.

Quando si specifica il`publicKeyPath`, IDT convalida la chiave pubblica del dispositivo quando stabilisce una connessione SSH al dispositivo in prova. Se questo valore non è specificato, IDT crea una connessione SSH, ma non convalida la chiave pubblica del dispositivo.

Ti consigliamo vivamente di specificare il percorso della chiave pubblica e di utilizzare un metodo sicuro per recuperare questa chiave pubblica. Per i client SSH standard basati sulla riga di comando, la chiave pubblica viene fornita nel file. `known_hosts` Se si specifica un file con chiave pubblica separato, questo file deve utilizzare lo stesso formato del `known_hosts` file, ovvero. *`ip-address key-type public-key`* Se sono presenti più voci con lo stesso indirizzo IP, la voce relativa al tipo di chiave utilizzato da IDT deve precedere le altre voci del file.

`connectivity.auth`

Informazioni di autenticazione per la connessione.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.auth.method`

Il metodo di autorizzazione utilizzato per accedere a un dispositivo con un determinato protocollo di connettività.

I valori supportati sono:

- `pki`
- `password`

`connectivity.auth.credentials`

Le credenziali utilizzate per l'autenticazione.

`connectivity.auth.credentials.password`

La password utilizzata per l'accesso al dispositivo da testare.

Questo valore si applica solo se `connectivity.auth.method` è impostato su `password`.

`connectivity.auth.credentials.privKeyPath`

Il percorso completo alla chiave privata utilizzata per accedere al dispositivo sottoposto a test.

Questo valore si applica solo se `connectivity.auth.method` è impostato su `pki`.

`connectivity.auth.credentials.user`

Il nome utente per l'accesso al dispositivo sottoposto a test.

Configura userdata.json

IDT per AWS IoT Greengrass V2 necessita inoltre di informazioni aggiuntive sulla posizione degli artefatti e del software di test. AWS IoT Greengrass

Devi fornire queste informazioni utilizzando il modello `userdata.json` situato in *<device_tester_extract_location>*`/configs/userdata.json`:

```
{
    "TempResourcesDirOnDevice": "/path/to/temp/folder",
    "InstallationDirRootOnDevice": "/path/to/installation/folder",
    "GreengrassNucleusZip": "/path/to/aws.greengrass.nucleus.zip",
    "PreInstalled": "yes/no",
    "GreengrassV2TokenExchangeRole": "custom-iam-role-name",
 "hsm": {
        "greengrassPkcsPluginJar": "/path/to/aws.greengrass.crypto.Pkcs11Provider-
latest.jar",
        "pkcs11ProviderLibrary": "/path/to/pkcs11-vendor-library",
        "slotId": "slot-id",
        "slotLabel": "slot-label",
        "slotUserPin": "slot-pin",
        "keyLabel": "key-label",
        "preloadedCertificateArn": "certificate-arn"
        "rootCA": "path/to/root-ca"
    }
}
```

Tutte le proprietà che contengono valori sono obbligatorie come descritto di seguito:

`TempResourcesDirOnDevice`

Il percorso completo di una cartella temporanea sul dispositivo in esame in cui archiviare gli artefatti del test. Assicurati che non siano necessarie le autorizzazioni sudo per scrivere in questa directory.

> **Note**
>
> IDT elimina il contenuto di questa cartella al termine dell'esecuzione di un test.

`InstallationDirRootOnDevice`

Il percorso completo di una cartella sul dispositivo in cui eseguire l'installazione. AWS IoT Greengrass Per PreInstalled Greengrass, questo è il percorso della directory di installazione di Greengrass.

È necessario impostare le autorizzazioni di file richieste per questa cartella. Esegui il comando seguente per ogni cartella nel percorso di installazione.

```
sudo chmod 755 folder-name
```

`GreengrassNucleusZip`

Il percorso completo del file Greengrass nucleus ZIP (`greengrass-nucleus-latest.zip`) sul computer host. Questo campo non è obbligatorio per i test con PreInstalled Greengrass.

> **Note**
>
> Per informazioni sulle versioni supportate di Greengrass nucleus for IDT, vedere. AWS IoT Greengrass Ultima versione IDT per AWS IoT Greengrass V2 Per scaricare la versione più recente del software Greengrass, consulta Scaricare il AWS IoT Greengrass software.

`PreInstalled`

Questa funzionalità è disponibile per IDT v4.5.8 e versioni successive solo su dispositivi Linux.

(Facoltativo) Quando il valore è *yes*, IDT assumerà che il percorso indicato sia `InstallationDirRootOnDevice` la directory in cui è installato Greengrass.

Per ulteriori informazioni su come installare Greengrass sul dispositivo, vedere. Installa il software AWS IoT Greengrass Core con provisioning automatico delle risorse Se l'installazione avviene con il provisioning manuale, includi il passaggio «Aggiungi l'AWS IoToggetto a un gruppo di oggetti nuovo o esistente» quando crei un AWS IoToggetto manualmente. IDT presuppone che l'oggetto e il gruppo di oggetti vengano creati durante la configurazione dell'installazione. Assicuratevi che questi valori si riflettano nel `effectiveConfig.yaml` file. IDT verifica la presenza del file `effectiveConfig.yaml` sotto`<InstallationDirRootOnDevice>/config/effectiveConfig.yaml`.

Per eseguire test con HSM, assicurati che il `aws.greengrass.crypto.Pkcs11Provider` campo sia aggiornato in. `effectiveConfig.yaml`

`GreengrassV2TokenExchangeRole`

(Facoltativo) Il ruolo IAM personalizzato che desideri utilizzare come ruolo di scambio di token che il dispositivo sottoposto a test assume per interagire con AWS le risorse.

> **Note**
>
> IDT utilizza questo ruolo IAM personalizzato invece di creare il ruolo di scambio di token predefinito durante l'esecuzione del test. Se utilizzi un ruolo personalizzato, puoi aggiornare le autorizzazioni IAM per l'utente di test per escludere l'`iamResourcesUpdate`istruzione che consente all'utente di creare ed eliminare ruoli e politiche IAM.

Per ulteriori informazioni sulla creazione di un ruolo IAM personalizzato come ruolo di scambio di token, consultaConfigura un ruolo di scambio di token personalizzato.

`hsm`

Questa funzionalità è disponibile per IDT v4.5.1 e versioni successive.

(Facoltativo) Le informazioni di configurazione per il test con un AWS IoT Greengrass Hardware Security Module (HSM). Altrimenti, la proprietà `hsm` dovrebbe essere omessa. Per ulteriori informazioni, consulta Integrazione della sicurezza hardware.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

> **Warning**
>
> La configurazione HSM può essere considerata un dato sensibile se il modulo di sicurezza hardware è condiviso tra IDT e un altro sistema. In questa situazione, è possibile evitare di proteggere questi valori di configurazione in testo semplice memorizzandoli in un AWS parametro Parameter Store e configurando IDT per SecureString recuperarli durante l'esecuzione del test. Per ulteriori informazioni, consultare ???

`hsm.greengrassPkcsPluginJar`

Il percorso completo del componente del provider PKCS #11 scaricato sul computer host IDT. AWS IoT Greengrassfornisce questo componente come file JAR che è possibile scaricare per specificarlo come plug-in di provisioning durante l'installazione. È possibile scaricare la versione più recente del file JAR del componente al seguente URL: https://d2s8p88vqu9w66.cloudfront.net/releases/Pkcs11Provider/aws.greengrass.crypto.Pkcs11Provider-latest.jar.

`hsm.pkcs11ProviderLibrary`

Il percorso completo della libreria PKCS #11 fornita dal fornitore del modulo di sicurezza hardware (HSM) per interagire con l'HSM.

`hsm.slotId`

L'ID dello slot utilizzato per identificare lo slot HSM in cui caricare la chiave e il certificato.

`hsm.slotLabel`

L'etichetta dello slot utilizzata per identificare lo slot HSM in cui caricare la chiave e il certificato.

`hsm.slotUserPin`

Il PIN utente utilizzato da IDT per autenticare il software AWS IoT Greengrass Core nell'HSM.

> **Note**
>
> Come procedura consigliata per la sicurezza, non utilizzate lo stesso PIN utente sui dispositivi di produzione.

`hsm.keyLabel`

L'etichetta utilizzata per identificare la chiave nel modulo hardware. Sia la chiave che il certificato devono utilizzare la stessa etichetta chiave.

`hsm.preloadedCertificateArn`

L'Amazon Resource Name (ARN) del certificato del dispositivo caricato nel AWS IoT cloud.

Devi aver precedentemente generato questo certificato utilizzando la chiave nell'HSM, importarlo nel tuo HSM e caricarlo sul cloud. AWS IoT Per informazioni sulla generazione e l'importazione del certificato, consulta la documentazione del tuo HSM.

È necessario caricare il certificato sullo stesso account e sulla stessa regione forniti in config.json. . Per ulteriori informazioni sul caricamento del certificato suAWS IoT, consulta Registrare manualmente un certificato client nella Guida per gli AWS IoT sviluppatori.

`hsm.rootCAPath`

(Facoltativo) Il percorso completo sulla macchina host IDT dell'autorità di certificazione (CA) principale che ha firmato il certificato. Questo è necessario se il certificato nell'HSM creato non è firmato dalla CA root di Amazon.

Recupera la configurazione da Parameter Store AWS

AWS IoTDevice Tester (IDT) include una funzionalità opzionale per recuperare i valori di configurazione dal AWSSystems Manager Parameter Store. AWS Parameter Store consente l'archiviazione sicura e crittografata delle configurazioni. Una volta configurato, IDT può recuperare i parametri da AWS Parameter Store anziché archiviarli in testo semplice all'interno del file. `userdata.json` Ciò è utile per tutti i dati sensibili che devono essere archiviati in modo crittografato, ad esempio: password, pin e altri segreti.

1. Per utilizzare questa funzionalità, è necessario aggiornare le autorizzazioni utilizzate nella creazione dell'utente IDT per consentire l' GetParameter azione sui parametri per cui IDT è configurato. Di seguito è riportato un esempio di dichiarazione di autorizzazione che può essere aggiunta all'utente IDT. Per ulteriori informazioni, consulta AWS Systems Manageruserguide.

```
{
       "Sid":"parameterStoreResources",
       "Effect": "Allow",
        "Action": [
```

```
            "ssm:GetParameter"
        ],
        "Resource": "arn:aws:ssm:*:*:parameter/IDT*"
}
```

L'autorizzazione di cui sopra è configurata per consentire il recupero di tutti i parametri il cui nome inizia con`IDT`, utilizzando il carattere jolly. `*` Dovresti personalizzarlo in base alle tue esigenze in modo che IDT abbia accesso al recupero di tutti i parametri configurati in base alla denominazione dei parametri che stai utilizzando.

2. È necessario memorizzare i valori di configurazione all'interno AWS di Paramater Store. Questa operazione può essere eseguita dalla AWS console o dalla AWS CLI. AWS Parameter Store consente di scegliere l'archiviazione crittografata o non crittografata. Per l'archiviazione di valori sensibili come segreti, password e pin, è necessario utilizzare l'opzione crittografata che è un tipo di parametro di. SecureString Per caricare un parametro utilizzando la AWS CLI, puoi usare il seguente comando:

```
aws ssm put-parameter --name IDT-example-name --value IDT-example-value --type
 SecureString
```

È possibile verificare che un parametro sia memorizzato utilizzando il comando seguente. (Facoltativo) Utilizzate il `--with-decryption` flag per recuperare un parametro decrittografatoSecureString .

```
aws ssm get-parameter --name IDT-example-name
```

L'utilizzo della AWS CLI caricherà il parametro nella AWS regione dell'utente CLI corrente e IDT recupererà i parametri dalla regione configurata in. `config.json` Per controllare la tua regione dalla AWS CLI, usa quanto segue:

```
aws configure get region
```

3. Una volta che hai un valore di configurazione nel AWS cloud, puoi aggiornare qualsiasi valore all'interno della configurazione IDT per recuperarlo dal cloud. AWS A tale scopo, utilizzate un segnaposto nella configurazione IDT del modulo `{{AWS.Parameter.parameter_name}}` per recuperare il parametro con quel nome dal Parameter Store. AWS

Ad esempio, supponiamo di voler utilizzare il `IDT-example-name` parametro del passaggio 2 come HSM KeyLabel nella configurazione HSM. Per fare ciò, puoi aggiornare il tuo come segue: `userdata.json`

```
"hsm": {
        "keyLabel": "{{AWS.Parameter.IDT-example-name}}",
        [...]
    }
```

IDT recupererà il valore di questo parametro in fase di esecuzione impostato allo Step `IDT-example-value` 2. Questa configurazione è simile all'impostazione `"keyLabel": "IDT-example-value"` ma, invece, tale valore viene archiviato come crittografato nel AWS Cloud.

Esegui la suite AWS IoT Greengrass di qualificazione

Dopo avere impostato la configurazione richiesta, puoi iniziare i test. Il runtime delle suite di test completa dipende dall'hardware. Per riferimento, per completare la suite di test completa su un Raspberry Pi 3B sono necessari circa 30 minuti.

Usa il `run-suite` comando seguente per eseguire una suite di test.

```
devicetester_[linux | mac | win]_x86-64 run-suite  \\
    --suite-id suite-id  \\
    --group-id group-id  \\
    --pool-id your-device-pool \\
    --test-id test-id  \\
    --update-idt y|n  \\
    --userdata userdata.json
```

Tutte le opzioni sono opzionali. Ad esempio, è possibile omettere `pool-id` se si dispone di un solo pool di dispositivi, ovvero un insieme di dispositivi identici, definito nel `device.json` file. In alternativa, è possibile omettere `suite-id` se si desidera eseguire l'ultima versione della suite di test nella cartella `tests`.

> **Note**
>
> IDT chiede se è disponibile online una versione più recente della suite di test. Per ulteriori informazioni, consulta the section called “Versioni della suite di test”.

Comandi di esempio per eseguire la suite di qualificazione

I seguenti esempi da riga di comando mostrano come eseguire i test di qualificazione per un pool di dispositivi. Per ulteriori informazioni su `run-suite` e altri comandi IDT, consulta the section called “Comandi IDT”.

Utilizzate il comando seguente per eseguire tutti i gruppi di test in una suite di test specificata. Il `list-suites` comando elenca le suite di test presenti nella `tests` cartella.

```
devicetester_[linux | mac | win]_x86-64 run-suite \
    --suite-id GGV2Q_1.0.0 \
    --pool-id <pool-id> \
    --userdata userdata.json
```

Utilizzate il comando seguente per eseguire un gruppo di test specifico in una suite di test. Il `list-groups` comando elenca i gruppi di test in una suite di test.

```
devicetester_[linux | mac | win]_x86-64 run-suite \
    --suite-id GGV2Q_1.0.0 \
    --group-id <group-id> \
    --pool-id <pool-id> \
    --userdata userdata.json
```

Utilizzate il comando seguente per eseguire un test case specifico in un gruppo di test.

```
devicetester_[linux | mac | win]_x86-64 run-suite \
    --group-id <group-id> \
    --test-id <test-id> \
    --userdata userdata.json
```

Utilizzate il comando seguente per eseguire più casi di test in un gruppo di test.

```
devicetester_[linux | mac | win]_x86-64 run-suite \
    --group-id <group-id> \
    --test-id <test-id1>,<test-id2>
    --userdata userdata.json
```

Usa il comando seguente per elencare tutti i casi di test in un gruppo di test.

```
devicetester_[linux | mac | win]_x86-64 list-test-cases --group-id <group-id>
```

Ti consigliamo di eseguire la suite completa di test di qualificazione, che esegue le dipendenze dei gruppi di test nell'ordine corretto. Se scegli di eseguire gruppi di test specifici, ti consigliamo di eseguire prima il gruppo di test del controllo delle dipendenze per assicurarti che tutte le dipendenze Greengrass siano installate prima di eseguire i relativi gruppi di test. Per esempio:

- Eseguire `coredependencies` prima dei gruppi di test di qualifica del core.

Comandi IDT per V2 AWS IoT Greengrass

I comandi IDT si trovano nella directory *<device-tester-extract-location>*`/bin`. Per eseguire una suite di test, fornite il comando nel seguente formato:

`help`

Elenca le informazioni sul comando specificato.

`list-groups`

Elenca i gruppi in una determinata suite di test.

`list-suites`

Elenca le suite di test disponibili.

`list-supported-products`

Elenca i prodotti supportati, in questo caso le versioni e le versioni della suite di test AWS IoT Greengrass per la versione IDT corrente.

`list-test-cases`

Elenca i casi di test in un determinato gruppo di test. È supportata la seguente opzione:

- `group-id`. Il gruppo di test da cercare. Questa opzione è obbligatoria e deve specificare un singolo gruppo.

`run-suite`

Esegue una suite di test in un determinato pool di dispositivi. Di seguito sono riportate alcune opzioni supportate:

- `suite-id`. La versione della suite di test da eseguire. Se non specificato, IDT utilizza la versione più recente nella cartella `tests`.
- `group-id`. I gruppi di test da eseguire, sotto forma di elenco separato da virgole. Se non specificato, IDT esegue tutti i gruppi di test appropriati nella suite di test a seconda delle

impostazioni configurate in. `device.json` IDT non esegue alcun gruppo di test che il dispositivo non supporta in base alle impostazioni configurate, anche se tali gruppi di test sono specificati nell'`group-id`elenco.

- `test-id`. I casi di test da eseguire, come elenco separato da virgole. Quando specificato, `group-id` deve specificare un singolo gruppo.
- `pool-id`. Il pool di dispositivi da testare. È necessario specificare un pool se nel file `device.json` sono stati definiti più pool di dispositivi.
- `stop-on-first-failure`. Configura IDT in modo che smetta di funzionare al primo errore. Utilizzate questa opzione per eseguire `group-id` il debug dei gruppi di test specificati. Non utilizzare questa opzione quando si esegue una suite di test completa per generare un rapporto di qualifica.
- `update-idt`. Imposta la risposta alla richiesta di aggiornamento di IDT. La `Y` risposta interrompe l'esecuzione del test se IDT rileva l'esistenza di una versione più recente. La `N` risposta continua l'esecuzione del test.
- `userdata`. Il percorso completo del `userdata.json` file che contiene informazioni sui percorsi degli artefatti di test. Questa opzione è obbligatoria per il `run-suite` comando. *`Il userdata.json file deve trovarsi nella directory devicetester_extract_location /devicetester_ggv2_ [win|mac|linux] / configs/.`*

Per ulteriori informazioni sulle opzioni `run-suite`, utilizzare l'opzione `help`:

```
devicetester_[linux | mac | win]_x86-64 run-suite -h
```

Informazioni su risultati e log

Questa sezione descrive come visualizzare e interpretare i log e i report dei risultati di IDT.

Per risolvere gli errori, consultaRisoluzione dei problemi IDT perAWS IoT GreengrassV2.

Visualizzazione dei risultati

Durante l'esecuzione, IDT scrive gli errori nella console, i file di log e i report di test. Al termine della suite di test di qualifica, IDT genera due report di test. Questi report si trovano in*`<device-tester-extract-location>`*`/results/`*`<execution-id>`*`/`. Entrambi i report acquisiscono i risultati dell'esecuzione della suite di test di qualificazione.

`awsiotdevicetester_report.xml`È il rapporto del test di qualificazione a cui invii il tuo dispositivoAWS per elencare il tuoAWS Partner dispositivo nel catalogo dei dispositivi. Il report contiene i seguenti elementi:

- La versione di IDT.
- La versione di AWS IoT Greengrass che è stata sottoposta a test.
- Il codice SKU e il nome del pool di dispositivi specificato nel file `device.json`.
- Le caratteristiche del pool di dispositivi specificato nel file `device.json`.
- Il riepilogo aggregato dei risultati dei test.
- Una suddivisione dei risultati dei test per librerie testate in base alle funzionalità del dispositivo, come accesso alle risorse locali, shadow e MQTT.

Il report `GGV2Q_Result.xml` è in formato XML JUnit. Puoi eseguire l'integrazione in piattaforme di integrazione e distribuzione continue come Jenkins, Bambù e così via. Il report contiene i seguenti elementi:

- Riepilogo aggregato dei risultati dei test.
- Analisi dei risultati dei test in base alla funzionalità AWS IoT Greengrass che è stata sottoposta a test.

InterpretazioneAWS IoT Device Tester dei risultati

La sezione dei report in `awsiotdevicetester_report.xml` o `awsiotdevicetester_report.xml` elenca i test eseguiti e i risultati.

Il primo tag XML`<testsuites>` contiene il riepilogo dell'esecuzione del test. Ad esempio:

```
<testsuites name="GGQ results" time="2299" tests="28" failures="0" errors="0"
 disabled="0">
```

Attributi utilizzati nel tag **`<testsuites>`**

`name`

Il nome della suite di test.

`time`

La durata, espressa in secondi, in cui la suite di qualificazione è stata eseguita.

`tests`

Il numero di test eseguiti.

`failures`

Il numero di test eseguiti ma non superati.

`errors`

Il numero di test che IDT non è riuscito a eseguire.

`disabled`

Ignora questo attributo. Non viene utilizzato.

Il file `awsiotdevicetester_report.xml` contiene un tag `<awsproduct>` con le informazioni relative al prodotto sottoposto a test e le caratteristiche del prodotto che sono state convalidate dopo l'esecuzione di una suite di test.

Attributi utilizzati nel tag **`<awsproduct>`**

`name`

Il nome del prodotto sottoposto a test.

`version`

La versione del prodotto sottoposto a test.

`features`

Le caratteristiche convalidate. Le caratteristiche contrassegnate come `required` sono necessarie per inviare la scheda per la qualifica. Il seguente frammento di codice mostra come questa informazione viene visualizzata nel file `awsiotdevicetester_report.xml`.

```
<name="aws-iot-greengrass-v2-core" value="supported" type="required"></feature>
```

Se non ci sono guasti o errori nei test per le caratteristiche richieste, il dispositivo soddisfa i requisiti tecnici per l'esecuzione di AWS IoT Greengrass e può interagire con i servizi AWS IoT. Se desideri

inserire il tuo dispositivo nel catalogo deiAWS Partner dispositivi, puoi utilizzare questo rapporto come prova di qualificazione.

In caso di esiti negativi o errori nei test, puoi identificare il test non riuscito esaminando i tag XML `<testsuites>`. I tag XML `<testsuite>` all'interno del tag `<testsuites>` mostrano il riepilogo dei risultati dei test per un gruppo di test. Ad esempio:

```
<testsuite name="combination" package="" tests="1" failures="0" time="161" disabled="0"
 errors="0" skipped="0">
```

Il formato è simile al tag `<testsuites>`, ma con un attributo `skipped` che non viene utilizzato e che è possibile ignorare. All'interno di ogni tag`<testsuite>` XML, ci sono`<testcase>` tag per ogni test eseguito per un gruppo di test. Ad esempio:

```
<testcase classname="Security Combination (IPD + DCM) Test Context" name="Security
 Combination IP Change Tests sec4_test_1: Should rotate server cert when IPD disabled
 and following changes are made:Add CIS conn info and Add another CIS conn info"
 attempts="1"></testcase>>
```

Attributi utilizzati nel tag **`<testcase>`**

`name`

Il nome del test.

`attempts`

Il numero di volte in cui IDT ha eseguito il test case.

Quando un test non riesce o si verifica un errore, i tag `<failure>` o `<error>` vengono aggiunti al tag `<testcase>` con informazioni per la risoluzione dei problemi. Ad esempio:

```
<testcase classname="mcu.Full_MQTT" name="AFQP_MQTT_Connect_HappyCase" attempts="1">
 <failure type="Failure">Reason for the test failure</failure>
 <error>Reason for the test execution error</error>
</testcase>
```

Visualizzazione dei registri di

IDT genera registri dalle esecuzioni dei test in*`<devicetester-extract-location>`*`/results/`*`<execution-id>`*`/logs`. Vengono generate due serie di log:

`test_manager.log`

Registri generati dal componente Test Manager diAWS IoT Device Tester (ad esempio, registri relativi alla configurazione, alla sequenza dei test e alla generazione di report).

<test-case-id>`.log (for example, lambdaDeploymentTest.log)`

Registri del test case all'interno del gruppo di test, inclusi i registri del dispositivo sottoposto a test. A partire da IDT v4.2.0, IDT raggruppa i log dei test per ogni test case in una cartella *<test-case-id >* separata all'interno della*<devicetester-extract-location>*`/results/`*<execution-id>*`/logs/`*<test-group-id>/* directory.

Usa IDT per sviluppare ed eseguire le tue suite di test

A partire da IDT v4.0.1, IDT per AWS IoT Greengrass V2 combina una configurazione di configurazione e un formato di risultati standardizzati con un ambiente di suite di test che consente di sviluppare suite di test personalizzate per i dispositivi e il software del dispositivo. Potete aggiungere test personalizzati per la vostra convalida interna o fornirli ai clienti per la verifica dei dispositivi.

Utilizza IDT per sviluppare ed eseguire suite di test personalizzate, come segue:

Per sviluppare suite di test personalizzate

- Crea suite di test con logica di test personalizzata per il dispositivo Greengrass che desideri testare.
- Fornisci a IDT le tue suite di test personalizzate per i test runner. Includi informazioni sulle configurazioni di impostazioni specifiche per le tue suite di test.

Per eseguire suite di test personalizzate

- Configura il dispositivo che desideri testare.
- Implementa le configurazioni delle impostazioni come richiesto dalle suite di test che desideri utilizzare.
- Usa IDT per eseguire le tue suite di test personalizzate.
- Visualizza i risultati dei test e i registri di esecuzione per i test eseguiti da IDT.

Scarica l'ultima versione di for AWS IoT Device TesterAWS IoT Greengrass

Scaricate l'ultima versione di IDT ed estraete il software in una posizione (*< device-tester-extract-location >*) del file system in cui disponete delle autorizzazioni di lettura/scrittura.

> **Note**
>
> IDT non supporta l'esecuzione da parte di più utenti da un percorso condiviso, ad esempio una directory NFS o una cartella condivisa di rete Windows. Si consiglia di estrarre il pacchetto IDT in un'unità locale ed eseguire il file binario IDT sulla workstation locale.
> In Windows esiste un limite di lunghezza del percorso di 260 caratteri. Se stai usando Windows, estrai IDT in una directory root come `C:\` o `D:\` per mantenere i percorsi entro il limite di 260 caratteri.

Flusso di lavoro per la creazione della suite

Le suite di test sono composte da tre tipi di file:

- File di configurazione che forniscono a IDT informazioni su come eseguire la suite di test.
- Esegui il test dei file eseguibili utilizzati da IDT per eseguire i test case.
- File aggiuntivi necessari per eseguire i test.

Completa i seguenti passaggi di base per creare test IDT personalizzati:

1. Crea file di configurazione per la tua suite di test.
2. Crea eseguibili per test case che contengano la logica di test per la tua suite di test.
3. Verifica e documenta le informazioni di configurazione richieste ai test runner per eseguire la suite di test.
4. Verifica che IDT sia in grado di eseguire la tua suite di test e produrre i risultati dei test come previsto.

Per creare rapidamente una suite personalizzata di esempio ed eseguirla, segui le istruzioni riportate inTutorial: crea ed esegui la suite di test IDT di esempio.

Per iniziare a creare una suite di test personalizzata in Python, vedi. Tutorial: Sviluppa una semplice suite di test IDT

Tutorial: crea ed esegui la suite di test IDT di esempio

Il AWS IoT Device Tester download include il codice sorgente per una suite di test di esempio. Puoi completare questo tutorial per creare ed eseguire la suite di test di esempio e capire come utilizzare IDT per AWS IoT Greengrass eseguire suite di test personalizzate.

In questo tutorial, completerai i seguenti passaggi:

1. Crea la suite di test di esempio
2. Usa IDT per eseguire la suite di test di esempio

Prerequisiti

Per completare questo tutorial, è necessario quanto segue:

- Requisiti del computer host
  - Ultima versione di AWS IoT Device Tester
  - Python 3.7 o successivo

    Per verificare la versione di Python installata sul tuo computer, esegui il seguente comando:

    ```
    python3 --version
    ```

    In Windows, se l'utilizzo di questo comando restituisce un errore, usalo `python --version` invece. Se il numero di versione restituito è 3.7 o superiore, esegui il comando seguente in un terminale Powershell da impostare `python3` come alias per il comando. `python`

    ```
    Set-Alias -Name "python3" -Value "python"
    ```

    Se non viene restituita alcuna informazione sulla versione o se il numero di versione è inferiore a 3.7, segui le istruzioni in Downloading Python per installare Python 3.7+. Per ulteriori informazioni, consulta la documentazione di Python.
  - urllib3

    Per verificare che `urllib3` sia installato correttamente, esegui il seguente comando:

    ```
    python3 -c 'import urllib3'
    ```

Se non `urllib3` è installato, esegui il seguente comando per installarlo:

```
python3 -m pip install urllib3
```

- Requisiti per il dispositivo
  - Un dispositivo con un sistema operativo Linux e una connessione di rete alla stessa rete del computer host.

    Ti consigliamo di utilizzare un Raspberry Pi con sistema operativo Raspberry Pi. Assicurati di aver configurato SSH sul tuo Raspberry Pi per connetterti in remoto ad esso.

Configura le informazioni sul dispositivo per IDT

Configura le informazioni sul dispositivo per consentire a IDT di eseguire il test. È necessario aggiornare il `device.json` modello che si trova nella *`<device-tester-extract-location>`*`/configs` cartella con le seguenti informazioni.

```
[
  {
    "id": "pool",
    "sku": "N/A",
    "devices": [
      {
        "id": "<device-id>",
        "connectivity": {
          "protocol": "ssh",
          "ip": "<ip-address>",
          "port": "<port>",
          "auth": {
            "method": "pki | password",
            "credentials": {
              "user": "<user-name>",
              "privKeyPath": "/path/to/private/key",
              "password": "<password>"
            }
          }
        }
      }
    ]
  }
```

```
]
```

Nell'`devices`oggetto, fornite le seguenti informazioni:

`id`

Un identificatore univoco definito dall'utente per il dispositivo.

`connectivity.ip`

L'indirizzo IP del dispositivo.

`connectivity.port`

Facoltativo. Il numero di porta da utilizzare per le connessioni SSH al dispositivo.

`connectivity.auth`

Informazioni di autenticazione per la connessione.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.auth.method`

Il metodo di autorizzazione utilizzato per accedere a un dispositivo con un determinato protocollo di connettività.

I valori supportati sono:

- `pki`
- `password`

`connectivity.auth.credentials`

Le credenziali utilizzate per l'autenticazione.

`connectivity.auth.credentials.user`

Il nome utente utilizzato per accedere al dispositivo.

`connectivity.auth.credentials.privKeyPath`

Il percorso completo della chiave privata utilizzata per accedere al dispositivo.

Questo valore si applica solo se `connectivity.auth.method` è impostato su `pki`.

`devices.connectivity.auth.credentials.password`

La password utilizzata per accedere al dispositivo.

Questo valore si applica solo se `connectivity.auth.method` è impostato su `password`.

> **Note**
>
> Specificare `privKeyPath` solo se `method` è impostato su `pki`.
> Specificare `password` solo se `method` è impostato su `password`.

Crea la suite di test di esempio

La *<device-tester-extract-location>*`/samples/python` cartella contiene file di configurazione di esempio, codice sorgente e IDT Client SDK che puoi combinare in una suite di test utilizzando gli script di build forniti. Il seguente albero di directory mostra la posizione di questi file di esempio:

```
<device-tester-extract-location>
### ...
### tests
### samples
#   ### ...
#   ### python
#       ### configuration
#       ### src
#       ### build-scripts
#           ### build.sh
#           ### build.ps1
### sdks
    ### ...
    ### python
        ### idt_client
```

Per creare la suite di test, esegui i seguenti comandi sul tuo computer host:

Windows

```
cd <device-tester-extract-location>/samples/python/build-scripts
```

```
./build.ps1
```

Linux, macOS, or UNIX

```
cd <device-tester-extract-location>/samples/python/build-scripts
./build.sh
```

In questo modo viene creata la suite di test di esempio nella `IDTSampleSuitePython_1.0.0` cartella all'interno della *<device-tester-extract-location>*`/tests` cartella. Esamina i file nella `IDTSampleSuitePython_1.0.0` cartella per capire come è strutturata la suite di test di esempio e per vedere vari esempi di eseguibili per test case e file JSON di configurazione dei test.

> **Note**
>
> La suite di test di esempio include il codice sorgente di Python. Non includete informazioni sensibili nel codice della suite di test.

Passaggio successivo: usa IDT per eseguire la suite di test di esempio che hai creato.

Usa IDT per eseguire la suite di test di esempio

Per eseguire la suite di test di esempio, esegui i seguenti comandi sul tuo computer host:

```
cd <device-tester-extract-location>/bin
./devicetester_[linux | mac | win_x86-64] run-suite --suite-id IDTSampleSuitePython
```

IDT esegue la suite di test di esempio e trasmette i risultati alla console. Al termine dell'esecuzione del test, vengono visualizzate le seguenti informazioni:

```
========== Test Summary ==========
Execution Time:         5s
Tests Completed:        4
Tests Passed:           4
Tests Failed:           0
Tests Skipped:          0
----------------------------------
Test Groups:
    sample_group:       PASSED
```

```
---------------------------------
Path to IoT Device Tester Report: /path/to/devicetester/
results/87e673c6-1226-11eb-9269-8c8590419f30/awsiotdevicetester_report.xml
Path to Test Execution Logs: /path/to/devicetester/
results/87e673c6-1226-11eb-9269-8c8590419f30/logs
Path to Aggregated JUnit Report: /path/to/devicetester/
results/87e673c6-1226-11eb-9269-8c8590419f30/IDTSampleSuitePython_Report.xml
```

Risoluzione dei problemi

Utilizza le seguenti informazioni per risolvere eventuali problemi relativi al completamento del tutorial.

Il test case non viene eseguito correttamente

Se il test non viene eseguito correttamente, IDT trasmette i log degli errori alla console per aiutarti a risolvere i problemi relativi all'esecuzione del test. Assicurati di soddisfare tutti i prerequisiti per questo tutorial.

Impossibile connettersi al dispositivo in prova

Verificare quanto segue:

- Il `device.json` file contiene l'indirizzo IP, la porta e le informazioni di autenticazione corretti.
- Puoi connetterti al tuo dispositivo tramite SSH dal tuo computer host.

Tutorial: Sviluppa una semplice suite di test IDT

Una suite di test combina quanto segue:

- Eseguibili di test che contengono la logica di test
- File di configurazione che descrivono la suite di test

Questo tutorial mostra come usare IDT per AWS IoT Greengrass sviluppare una suite di test Python che contenga un singolo test case. In questo tutorial, completerai i seguenti passaggi:

1. Crea una directory per la suite di test
2. Crea file di configurazione
3. Crea l'eseguibile del test case
4. Esegui la suite di test

Prerequisiti

Per completare questo tutorial, è necessario quanto segue:

- Requisiti del computer host
  - Ultima versione di AWS IoT Device Tester
  - Python 3.7 o successivo

    Per verificare la versione di Python installata sul tuo computer, esegui il seguente comando:

    ```
    python3 --version
    ```

    In Windows, se l'utilizzo di questo comando restituisce un errore, usalo `python --version` invece. Se il numero di versione restituito è 3.7 o superiore, esegui il comando seguente in un terminale Powershell da impostare `python3` come alias per il comando. `python`

    ```
    Set-Alias -Name "python3" -Value "python"
    ```

    Se non viene restituita alcuna informazione sulla versione o se il numero di versione è inferiore a 3.7, segui le istruzioni in Downloading Python per installare Python 3.7+. Per ulteriori informazioni, consulta la documentazione di Python.
  - urllib3

    Per verificare che `urllib3` sia installato correttamente, esegui il seguente comando:

    ```
    python3 -c 'import urllib3'
    ```

    Se non `urllib3` è installato, esegui il seguente comando per installarlo:

    ```
    python3 -m pip install urllib3
    ```
- Requisiti per il dispositivo
  - Un dispositivo con un sistema operativo Linux e una connessione di rete alla stessa rete del computer host.

    Ti consigliamo di utilizzare un Raspberry Pi con sistema operativo Raspberry Pi. Assicurati di aver configurato SSH sul tuo Raspberry Pi per connetterti in remoto ad esso.

Crea una directory per la suite di test

IDT separa logicamente i casi di test in gruppi di test all'interno di ciascuna suite di test. Ogni test case deve essere all'interno di un gruppo di test. Per questo tutorial, crea una cartella chiamata `MyTestSuite_1.0.0` e crea il seguente albero di directory all'interno di questa cartella:

```
MyTestSuite_1.0.0
### suite
    ### myTestGroup
        ### myTestCase
```

Crea file di configurazione

La tua suite di test deve contenere i seguenti file di configurazione richiesti:

File di configurazione richiesti

`suite.json`

Contiene informazioni sulla suite di test. Per informazioni, consulta Configura suite.json.

`group.json`

Contiene informazioni su un gruppo di test. È necessario creare un `group.json` file per ogni gruppo di test nella suite di test. Per informazioni, consulta Configura group.json.

`test.json`

Contiene informazioni su un test case. È necessario creare un `test.json` file per ogni test case nella suite di test. Per informazioni, consulta Configura test.json.

1. Nella `MyTestSuite_1.0.0/suite` cartella, create un `suite.json` file con la seguente struttura:

   ```
   {
       "id": "MyTestSuite_1.0.0",
       "title": "My Test Suite",
       "details": "This is my test suite.",
       "userDataRequired": false
   }
   ```

2. Nella `MyTestSuite_1.0.0/myTestGroup` cartella, create un `group.json` file con la seguente struttura:

```
{
    "id": "MyTestGroup",
    "title": "My Test Group",
    "details": "This is my test group.",
    "optional": false
}
```

3. Nella `MyTestSuite_1.0.0/myTestGroup/myTestCase` cartella, create un `test.json` file con la seguente struttura:

```
{
    "id": "MyTestCase",
    "title": "My Test Case",
    "details": "This is my test case.",
    "execution": {
        "timeout": 300000,
        "linux": {
            "cmd": "python3",
            "args": [
                "myTestCase.py"
            ]
        },
        "mac": {
            "cmd": "python3",
            "args": [
                "myTestCase.py"
            ]
        },
        "win": {
            "cmd": "python3",
            "args": [
                "myTestCase.py"
            ]
        }
    }
}
```

L'albero delle cartelle della `MyTestSuite_1.0.0` cartella dovrebbe ora avere il seguente aspetto:

```
MyTestSuite_1.0.0
### suite
```

```
    ### suite.json
    ### myTestGroup
        ### group.json
        ### myTestCase
            ### test.json
```

Scarica l'SDK del client IDT

Utilizzi l'SDK del client IDT per consentire a IDT di interagire con il dispositivo sottoposto a test e di riportare i risultati del test. Per questo tutorial, utilizzerai la versione Python dell'SDK.

Dalla *<device-tester-extract-location>*`/sdks/python/` cartella, copia la `idt_client` cartella nella tua `MyTestSuite_1.0.0/suite/myTestGroup/myTestCase` cartella.

Per verificare che l'SDK sia stato copiato correttamente, esegui il comando seguente.

```
cd MyTestSuite_1.0.0/suite/myTestGroup/myTestCase
python3 -c 'import idt_client'
```

Crea l'eseguibile del test case

Gli eseguibili del test case contengono la logica di test che si desidera eseguire. Una suite di test può contenere più eseguibili di test case. Per questo tutorial, creerai un solo eseguibile di test case.

1. Crea il file della suite di test.

   Nella `MyTestSuite_1.0.0/suite/myTestGroup/myTestCase` cartella, crea un `myTestCase.py` file con il seguente contenuto:

   ```
   from idt_client import *

   def main():
       # Use the client SDK to communicate with IDT
       client = Client()

   if __name__ == "__main__":
       main()
   ```

2. Utilizza le funzioni dell'SDK del client per aggiungere la seguente logica di test al tuo `myTestCase.py` file:

   a. Esegui un comando SSH sul dispositivo sottoposto a test.

```
from idt_client import *

def main():
    # Use the client SDK to communicate with IDT
    client = Client()

    # Create an execute on device request
    exec_req = ExecuteOnDeviceRequest(ExecuteOnDeviceCommand("echo 'hello
 world'"))

    # Run the command
    exec_resp = client.execute_on_device(exec_req)

    # Print the standard output
    print(exec_resp.stdout)

if __name__ == "__main__":
    main()
```

b. Invia il risultato del test a IDT.

```
from idt_client import *

def main():
    # Use the client SDK to communicate with IDT
    client = Client()

    # Create an execute on device request
    exec_req = ExecuteOnDeviceRequest(ExecuteOnDeviceCommand("echo 'hello
 world'"))

    # Run the command
    exec_resp = client.execute_on_device(exec_req)

    # Print the standard output
    print(exec_resp.stdout)

    # Create a send result request
    sr_req = SendResultRequest(TestResult(passed=True))

    # Send the result
    client.send_result(sr_req)
```

```
if __name__ == "__main__":
    main()
```

Configura le informazioni sul dispositivo per IDT

Configura le informazioni sul dispositivo per consentire a IDT di eseguire il test. È necessario aggiornare il `device.json` modello che si trova nella *<device-tester-extract-location>*`/configs` cartella con le seguenti informazioni.

```
[
  {
    "id": "pool",
    "sku": "N/A",
    "devices": [
      {
        "id": "<device-id>",
        "connectivity": {
          "protocol": "ssh",
          "ip": "<ip-address>",
          "port": "<port>",
          "auth": {
            "method": "pki | password",
            "credentials": {
              "user": "<user-name>",
              "privKeyPath": "/path/to/private/key",
              "password": "<password>"
            }
          }
        }
      }
    ]
  }
]
```

Nell'`devices`oggetto, fornite le seguenti informazioni:

`id`

Un identificatore univoco definito dall'utente per il dispositivo.

`connectivity.ip`

L'indirizzo IP del dispositivo.

`connectivity.port`

Facoltativo. Il numero di porta da utilizzare per le connessioni SSH al dispositivo.

`connectivity.auth`

Informazioni di autenticazione per la connessione.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.auth.method`

Il metodo di autorizzazione utilizzato per accedere a un dispositivo con un determinato protocollo di connettività.

I valori supportati sono:

- `pki`
- `password`

`connectivity.auth.credentials`

Le credenziali utilizzate per l'autenticazione.

`connectivity.auth.credentials.user`

Il nome utente utilizzato per accedere al dispositivo.

`connectivity.auth.credentials.privKeyPath`

Il percorso completo della chiave privata utilizzata per accedere al dispositivo.

Questo valore si applica solo se `connectivity.auth.method` è impostato su `pki`.

`devices.connectivity.auth.credentials.password`

La password utilizzata per accedere al dispositivo.

Questo valore si applica solo se `connectivity.auth.method` è impostato su `password`.

> **Note**
>
> Specificare `privKeyPath` solo se `method` è impostato su `pki`.

Specificare `password` solo se `method` è impostato su `password`.

Esegui la suite di test

Dopo aver creato la suite di test, devi assicurarti che funzioni come previsto. Completa i seguenti passaggi per eseguire la suite di test con il pool di dispositivi esistente a tale scopo.

1. Copia la tua `MyTestSuite_1.0.0` cartella in*<device-tester-extract-location>*/`tests`.
2. Esegui i comandi seguenti:

```
cd <device-tester-extract-location>/bin
./devicetester_[linux | mac | win_x86-64] run-suite --suite-id MyTestSuite
```

IDT esegue la tua suite di test e trasmette i risultati alla console. Al termine dell'esecuzione del test, vengono visualizzate le seguenti informazioni:

```
time="2020-10-19T09:24:47-07:00" level=info msg=Using pool: pool
time="2020-10-19T09:24:47-07:00" level=info msg=Using test suite "MyTestSuite_1.0.0"
 for execution
time="2020-10-19T09:24:47-07:00" level=info msg=b'hello world\n'
 suiteId=MyTestSuite groupId=myTestGroup testCaseId=myTestCase deviceId=my-device
 executionId=9a52f362-1227-11eb-86c9-8c8590419f30
time="2020-10-19T09:24:47-07:00" level=info msg=All tests finished.
 executionId=9a52f362-1227-11eb-86c9-8c8590419f30
time="2020-10-19T09:24:48-07:00" level=info msg=

========== Test Summary ==========
Execution Time:         1s
Tests Completed:        1
Tests Passed:           1
Tests Failed:           0
Tests Skipped:          0
----------------------------------
Test Groups:
    myTestGroup:        PASSED
----------------------------------
Path to IoT Device Tester Report: /path/to/devicetester/
results/9a52f362-1227-11eb-86c9-8c8590419f30/awsiotdevicetester_report.xml
```

```
Path to Test Execution Logs: /path/to/devicetester/
results/9a52f362-1227-11eb-86c9-8c8590419f30/logs
Path to Aggregated JUnit Report: /path/to/devicetester/
results/9a52f362-1227-11eb-86c9-8c8590419f30/MyTestSuite_Report.xml
```

Risoluzione dei problemi

Utilizza le seguenti informazioni per risolvere eventuali problemi relativi al completamento del tutorial.

Il test case non viene eseguito correttamente

Se il test non viene eseguito correttamente, IDT trasmette i log degli errori alla console per aiutarti a risolvere i problemi relativi all'esecuzione del test. Prima di controllare i log degli errori, verifica quanto segue:

- L'SDK del client IDT si trova nella cartella corretta, come descritto in questo passaggio.
- Soddisfi tutti i prerequisiti per questo tutorial.

Impossibile connettersi al dispositivo in prova

Verificare quanto segue:

- Il `device.json` file contiene l'indirizzo IP, la porta e le informazioni di autenticazione corretti.
- Puoi connetterti al tuo dispositivo tramite SSH dal tuo computer host.

Crea file di configurazione della suite di test IDT

Questa sezione descrive i formati in cui create i file di configurazione da includere quando scrivete una suite di test personalizzata.

File di configurazione richiesti

`suite.json`

Contiene informazioni sulla suite di test. Per informazioni, consulta Configura suite.json.

`group.json`

Contiene informazioni su un gruppo di test. È necessario creare un `group.json` file per ogni gruppo di test nella suite di test. Per informazioni, consulta Configura group.json.

`test.json`

Contiene informazioni su un test case. È necessario creare un `test.json` file per ogni test case nella suite di test. Per informazioni, consulta Configura test.json.

File di configurazione opzionali

`test_orchestrator.yaml` o `state_machine.json`

Definisce come vengono eseguiti i test quando IDT esegue la suite di test. Configura test_orchestrator.yamlSse.

> **Note**
>
> A partire da IDT v4.5.1, si utilizza il `test_orchestrator.yaml` file per definire il flusso di lavoro del test. Nelle versioni precedenti di IDT, si utilizza il file. `state_machine.json` Per informazioni sulla macchina a stati, vedereConfigurazione della macchina a stato IDT.

`userdata_schema.json`

Definisce lo schema per il `userdata.json`file che i test runner possono includere nella configurazione delle impostazioni. Il `userdata.json` file viene utilizzato per tutte le informazioni di configurazione aggiuntive necessarie per eseguire il test ma che non sono presenti nel `device.json` file. Per informazioni, consulta Configura userdata_schema.json.

I file di configurazione vengono inseriti nel file *<custom-test-suite-folder>* come illustrato qui.

```
<custom-test-suite-folder>
### suite
    ### suite.json
    ### test_orchestrator.yaml
    ### userdata_schema.json
    ### <test-group-folder>
        ### group.json
        ### <test-case-folder>
            ### test.json
```

Configura suite.json

Il `suite.json` file imposta le variabili di ambiente e determina se i dati utente sono necessari per eseguire la suite di test. Utilizzate il seguente modello per configurare il *<custom-test-suite-folder>*`/suite/suite.json` file:

```
{
    "id": "<suite-name>_<suite-version>",
    "title": "<suite-title>",
    "details": "<suite-details>",
    "userDataRequired": true | false,
    "environmentVariables": [
        {
            "key": "<name>",
            "value": "<value>",
        },
        ...
        {
            "key": "<name>",
            "value": "<value>",
        }
    ]
}
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`id`

Un ID univoco definito dall'utente per la suite di test. Il valore di `id` deve corrispondere al nome della cartella della suite di test in cui si trova il `suite.json` file. Il nome e la versione della suite devono inoltre soddisfare i seguenti requisiti:

- *<suite-name>*non può contenere caratteri di sottolineatura.
- *<suite-version>*è indicato come*x.x.x*, dove x è un numero.

L'ID viene visualizzato nei rapporti di test generati da IDT.

`title`

Un nome definito dall'utente per il prodotto o la funzionalità testata da questa suite di test. Il nome viene visualizzato nella CLI IDT per i test runner.

`details`

Una breve descrizione dello scopo della suite di test.

`userDataRequired`

Definisce se i test runner devono includere informazioni personalizzate in un `userdata.json` file. Se imposti questo valore su`true`, devi anche includere il `userdata_schema.json`file nella cartella della suite di test.

`environmentVariables`

Facoltativo. Una serie di variabili di ambiente da impostare per questa suite di test.

`environmentVariables.key`

Il nome della variabile di ambiente.

`environmentVariables.value`

Il valore della variabile di ambiente.

Configura group.json

Il `group.json` file definisce se un gruppo di test è obbligatorio o facoltativo. Utilizzate il seguente modello per configurare il *`<custom-test-suite-folder>`*`/suite/`*`<test-group>`*`/` `group.json` file:

```
{
    "id": "<group-id>",
    "title": "<group-title>",
    "details": "<group-details>",
    "optional": true | false,
}
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`id`

Un ID univoco definito dall'utente per il gruppo di test. Il valore di `id` deve corrispondere al nome della cartella del gruppo di test in cui si trova il `group.json` file e non può contenere caratteri di sottolineatura ()_. L'ID viene utilizzato nei report di test generati da IDT.

`title`

Un nome descrittivo per il gruppo di test. Il nome viene visualizzato nella CLI IDT per i test runner.

`details`

Una breve descrizione dello scopo del gruppo di test.

`optional`

Facoltativo. Imposta `true` per visualizzare questo gruppo di test come gruppo opzionale dopo che IDT ha terminato l'esecuzione dei test richiesti. Il valore predefinito è `false`.

Configura test.json

Il `test.json` file determina gli eseguibili del test case e le variabili di ambiente utilizzate da un test case. Per ulteriori informazioni sulla creazione di eseguibili per i test case, vedere. Crea file eseguibili per test case IDT

Utilizzate il seguente modello per configurare il *<custom-test-suite-folder>*/suite/*<test-group>*/*<test-case>*/test.json file:

```
{
    "id": "<test-id>",
    "title": "<test-title>",
    "details": "<test-details>",
    "requireDUT": true | false,
    "requiredResources": [
        {
            "name": "<resource-name>",
            "features": [
                {
                    "name": "<feature-name>",
                    "version": "<feature-version>",
                    "jobSlots": <job-slots>
                }
            ]
        }
    ],
    "execution": {
        "timeout": <timeout>,
        "mac": {
            "cmd": "/path/to/executable",
            "args": [
```

```
                "<argument>"
            ],
        },
        "linux": {
            "cmd": "/path/to/executable",
            "args": [
                "<argument>"
            ],
        },
        "win": {
            "cmd": "/path/to/executable",
            "args": [
                "<argument>"
            ]
        }
    },
    "environmentVariables": [
        {
            "key": "<name>",
            "value": "<value>",
        }
    ]
}
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`id`

Un ID univoco definito dall'utente per il test case. Il valore di `id` deve corrispondere al nome della cartella del test case in cui si trova il `test.json` file e non può contenere caratteri di sottolineatura ()_. L'ID viene utilizzato nei report di test generati da IDT.

`title`

Un nome descrittivo per il test case. Il nome viene visualizzato nella CLI IDT per i test runner.

`details`

Una breve descrizione dello scopo del test case.

`requireDUT`

Facoltativo. Imposta `true` se è necessario un dispositivo per eseguire questo test, altrimenti imposta su`false`. Il valore predefinito è `true`. I test runner configureranno i dispositivi che useranno per eseguire il test nel proprio `device.json` file.

`requiredResources`

Facoltativo. Un array che fornisce informazioni sui dispositivi di risorse necessari per eseguire questo test.

`requiredResources.name`

Il nome univoco da assegnare al dispositivo di risorse durante l'esecuzione di questo test.

`requiredResources.features`

Una serie di funzionalità del dispositivo di risorse definite dall'utente.

`requiredResources.features.name`

Il nome della funzionalità. La funzionalità del dispositivo per cui si desidera utilizzare questo dispositivo. Questo nome viene confrontato con il nome della funzionalità fornito dal test runner nel `resource.json` file.

`requiredResources.features.version`

Facoltativo. La versione della funzionalità. Questo valore viene confrontato con la versione della funzionalità fornita dal test runner nel `resource.json` file. Se non viene fornita una versione, la funzionalità non viene verificata. Se non è richiesto un numero di versione per la funzionalità, lascia vuoto questo campo.

`requiredResources.features.jobSlots`

Facoltativo. Il numero di test simultanei che questa funzionalità può supportare. Il valore predefinito è `1`. Se desideri che IDT utilizzi dispositivi distinti per le singole funzionalità, ti consigliamo di impostare questo valore su. 1

`execution.timeout`

La quantità di tempo (in millisecondi) che IDT attende prima che il test finisca. Per ulteriori informazioni sull'impostazione di questo valore, vedere. Crea file eseguibili per test case IDT

`execution.os`

Gli eseguibili del test case da eseguire in base al sistema operativo del computer host che esegue IDT. I valori supportati sono `linux`, `mac` e `win`.

`execution.os.cmd`

Il percorso dell'eseguibile del test case che si desidera eseguire per il sistema operativo specificato. Questa posizione deve trovarsi nel percorso di sistema.

`execution.os.args`

Facoltativo. Gli argomenti da fornire per eseguire l'eseguibile del test case.

`environmentVariables`

Facoltativo. Una serie di variabili di ambiente impostate per questo test case.

`environmentVariables.key`

Il nome della variabile di ambiente.

`environmentVariables.value`

Il valore della variabile di ambiente.

> **Note**
>
> Se specificate la stessa variabile di ambiente nel `test.json` file e nel `suite.json` file, il valore nel `test.json` file ha la precedenza.

Configura test_orchestrator.yaml

Un orchestrator di test è un costrutto che controlla il flusso di esecuzione della suite di test. Determina lo stato iniziale di una suite di test, gestisce le transizioni di stato in base a regole definite dall'utente e continua la transizione attraverso tali stati fino a raggiungere lo stato finale.

Se la vostra suite di test non include un orchestratore di test definito dall'utente, IDT genererà un orchestratore di test per voi.

L'orchestrator di test predefinito svolge le seguenti funzioni:

- Fornisce ai test runner la possibilità di selezionare ed eseguire gruppi di test specifici, anziché l'intera suite di test.
- Se non sono selezionati gruppi di test specifici, esegue ogni gruppo di test nella suite di test in ordine casuale.
- Genera report e stampa un riepilogo della console che mostra i risultati dei test per ogni gruppo di test e test case.

Per ulteriori informazioni sul funzionamento dell'IDT test orchestrator, consulta. Configurazione dell'orchestratore di test IDT

Configura userdata_schema.json

Il `userdata_schema.json` file determina lo schema in cui i test runner forniscono i dati degli utenti. I dati utente sono necessari se la suite di test richiede informazioni che non sono presenti nel `device.json` file. Ad esempio, i test potrebbero richiedere credenziali di rete Wi-Fi, porte aperte specifiche o certificati che un utente deve fornire. Queste informazioni possono essere fornite a IDT come parametro di input chiamato`userdata`, il cui valore è un `userdata.json` file, che gli utenti creano nella propria *`<device-tester-extract-location>`*`/config` cartella. Il formato del `userdata.json` file si basa sul `userdata_schema.json` file incluso nella suite di test.

Per indicare che i test runner devono fornire un `userdata.json` file:

1. Nel `suite.json` file, imposta su`userDataRequired`. `true`
2. Nel tuo*`<custom-test-suite-folder>`*, crea un `userdata_schema.json` file.
3. Modifica il `userdata_schema.json` file per creare uno schema JSON IETF Draft v4 valido.

Quando IDT esegue la suite di test, legge automaticamente lo schema e lo usa per convalidare il `userdata.json` file fornito dal test runner. Se valido, il contenuto del `userdata.json` file è disponibile sia nel contesto IDT che nel contesto del test orchestrator.

Configurazione dell'orchestratore di test IDT

A partire da IDT v4.5.1, IDT include un nuovoorchestratore di testcomponente. L'orchestratore di test è un componente IDT che controlla il flusso di esecuzione della suite di test e genera il report di test dopo che IDT ha terminato l'esecuzione di tutti i test. L'orchestratore di test determina la selezione dei test e l'ordine in cui i test vengono eseguiti in base a regole definite dall'utente.

Se la suite di test non include un orchestratore di test definito dall'utente, IDT genererà un orchestratore di test per te.

Il test orchestrator predefinito esegue le seguenti funzioni:

- Fornisce ai test runner la possibilità di selezionare ed eseguire gruppi di test specifici, anziché l'intera suite di test.
- Se non sono selezionati gruppi di test specifici, esegue tutti i gruppi di test nella suite di test in un ordine casuale.
- Genera report e stampa un riepilogo della console che mostra i risultati del test per ciascun gruppo di test e test case.

L'orchestratore di test sostituisce l'orchestratore di test IDT. Si consiglia vivamente di utilizzare l'orchestratore di test per sviluppare le suite di test anziché l'orchestratore di test IDT. L'orchestratore di test fornisce le seguenti funzionalità migliorate:

- Utilizza un formato dichiarativo rispetto al formato imperativo utilizzato dalla macchina a stato IDT. Ciò consente dispecificarequali test vuoi eseguire equandovuoi gestirli.
- Gestisce la gestione di gruppi specifici, la generazione di report, la gestione degli errori e il monitoraggio dei risultatiin modo che non sia richiestoper gestire manualmente queste azioni.
- Utilizza il formato YAML, che supporta i commenti per impostazione predefinita.
- Richiede80 percentomeno spazio su disco rispetto all'orchestratore di test per definire lo stesso flusso di lavoro.
- Aggiunge la convalida pre-test per verificare che la definizione del flusso di lavoro non contenga ID di test errati o dipendenze circolari.

Formato orchestrator di test

Puoi utilizzare il seguente modello per configurarne*<custom-test-suite-folder>*/suite/test_orchestrator.yamlfile:

```
Aliases:
  string: context-expression

ConditionalTests:
  - Condition: context-expression
    Tests:
      - test-descriptor

Order:
  - - group-descriptor
    - group-descriptor

Features:
  - Name: feature-name
    Value: support-description
    Condition: context-expression
    Tests:
        - test-descriptor
    OneOfTests:
        - test-descriptor
```

```
    IsRequired: boolean
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`Aliases`

Facoltativo. Stringhe definite dall'utente che mappano alle espressioni di contesto. Gli alias consentono di generare nomi descrittivi peridentificare le espressioni di contesto nella configurazione dell'orchestratore di test. Ciò è particolarmente utile in fase di creazione di espressioni o espressioni di contesto complesse che si utilizzano in più posizioni.

È possibile utilizzare le espressioni di contesto per memorizzare query di contesto che consentono di accedere ai dati da altre configurazioni IDT. Per ulteriori informazioni, consultare Accesso ai dati nel contesto.

Example Esempio

```
Aliases:
    FizzChosen: "'{{$pool.features[?(@.name == 'Fizz')].value[0]}}' == 'yes'"
    BuzzChosen: "'{{$pool.features[?(@.name == 'Buzz')].value[0]}}' == 'yes'"
    FizzBuzzChosen: "'{{$aliases.FizzChosen}}' && '{{$aliases.BuzzChosen}}'"
```

`ConditionalTests`

Facoltativo. Un elenco di condizioni e i casi di test corrispondenti che vengono eseguiti quando ogni condizione è soddisfatta. Ogni condizione può avere più casi di test; tuttavia, è possibile assegnare un determinato test case a una sola condizione.

Per impostazione predefinita, IDT esegue qualsiasi test case non assegnato a una condizione in questo elenco. Se non specificate questa sezione, IDT esegue tutti i gruppi di test nella suite di test.

Ogni articolo nel`ConditionalTests`include i seguenti parametri:

`Condition`

Un'espressione di contesto che restituiscebooleanovalue. Se il valore valutato è true, IDT esegue i test case specificati nel`Tests`Parametro .

`Tests`

L'elenco dei descrittori di test.

Ogni descrittore di test utilizza l'ID del gruppo di test e uno o più ID del test case per identificare i singoli test da eseguire da uno specifico gruppo di test. Il descrittore di test utilizza il seguente formato:

```
GroupId: group-id
CaseIds: [test-id, test-id] # optional
```

Example Esempio

L'esempio seguente utilizza espressioni di contesto generiche che è possibile definire come`Aliases`.

```
ConditionalTests:
    - Condition: "{{$aliases.Condition1}}"
      Tests:
          - GroupId: A
          - GroupId: B
    - Condition: "{{$aliases.Condition2}}"
      Tests:
          - GroupId: D
    - Condition: "{{$aliases.Condition1}} || {{$aliases.Condition2}}"
      Tests:
          - GroupId: C
```

In base alle condizioni definite, IDT seleziona i gruppi di test come segue:

- Se`Condition1`è vero, IDT esegue i test nei gruppi di test A, B e C.
- Se`Condition2`è vero, IDT esegue i test nei gruppi di test C e D.

`Order`

Facoltativo. L'ordine di esecuzione dei test. È possibile specificare l'ordine di prova a livello di gruppo di test. Se non si specifica questa sezione, IDT esegue tutti i gruppi di test applicabili in ordine casuale. Il valore di`Order`è un elenco di elenchi di descrittori di gruppo. Qualsiasi gruppo di test in cui non inserisci`Order`, può essere eseguito in parallel con qualsiasi altro gruppo di test elencato.

Ogni elenco di descrittori di gruppo contiene uno dei più descrittori di gruppo e identifica l'ordine in cui eseguire i gruppi specificati in ciascun descrittore. Puoi utilizzare i formati seguenti:

- *`group-id`*L'ID del gruppo di un gruppo di test esistente.

- `[group-id, group-id]`—Elenco dei gruppi di test che possono essere eseguiti in qualsiasi ordine l'uno rispetto all'altro.
- `"*"`—Carattere jolly. Equivale all'elenco di tutti i gruppi di test che non sono già specificati nell'elenco dei descrittori di gruppo corrente.

Il valore di`Order`devono inoltre soddisfare i seguenti requisiti:

- Gli ID del gruppo di test specificati in un descrittore di gruppo devono esistere nella suite di test.
- Ogni elenco dei descrittori di gruppo deve includere almeno un gruppo di test.
- Ogni elenco dei descrittori di gruppo deve contenere ID di gruppo univoci. Non è possibile ripetere un ID gruppo di test all'interno di singoli descrittori di gruppo.
- Un elenco di descrittori di gruppo può contenere al massimo un descrittore di gruppi jolly. Il descrittore del gruppo jolly deve essere il primo o l'ultimo elemento dell'elenco.

Example Esempi

Per una suite di test che contiene gruppi di test A, B, C, D ed E, l'elenco seguente di esempi mostra diversi modi per specificare che IDT deve prima eseguire il gruppo di test A, quindi eseguire il gruppo di test B e quindi eseguire i gruppi di test C, D ed E in qualsiasi ordine.

- ```
  Order:
      - - A
        - B
        - [C, D, E]
  ```
- ```
  Order:
      - - A
        - B
        - "*"
  ```
- ```
  Order:
      - - A
        - B

      - - B
        - C

      - - B
        - D

      - - B
  ```
  ```

```
      - E
```

`Features`

Facoltativo. L'elenco delle caratteristiche del prodotto che si desidera aggiungere IDT a la`awsiotdevicetester_report.xml`file. Se non specifichi questa sezione, IDT non aggiungerà alcuna funzionalità del prodotto al report.

Una funzione di prodotto è costituita da informazioni definite dall'utente su criteri specifici che un dispositivo potrebbe soddisfare. Ad esempio, la funzionalità del prodotto MQTT può indicare che il dispositivo pubblica correttamente i messaggi MQTT. Nello stato`awsiotdevicetester_report.xml`, le caratteristiche del prodotto sono impostate come`supported`,`not-supported`o un valore personalizzato definito dall'utente, in base al superamento dei test specificati.

Ogni articolo nel`Features`elenco comprende i seguenti parametri:

`Name`

Il nome della funzionalità.

`Value`

Facoltativo. Valore personalizzato che si desidera utilizzare nel report anziché`supported`. Se questo valore non è specificato, IDT basato imposta il valore della feature su`supported`o`not-supported`in base ai risultati dei test Se si verifica la stessa funzione con condizioni diverse, è possibile utilizzare un valore personalizzato per ogni istanza di tale feature nella`Features`list e IDT concatena i valori delle feature per le condizioni supportate. Per ulteriori informazioni, consulta

`Condition`

Un'espressione di contesto che restituisce`boolean`o`value`. Se il valore valutato è true, IDT aggiunge la funzionalità al report di test dopo aver terminato l'esecuzione della suite di test. Se il valore valutato è falso, il test non viene incluso nel report.

`Tests`

Facoltativo. L'elenco dei descrittori di test. Tutti i test specificati in questo elenco devono essere passati per supportare la funzionalità.

Ogni descrittore di test in questo elenco utilizza l'ID del gruppo di test e uno o più ID del test case per identificare i singoli test da eseguire da uno specifico gruppo di test. Il descrittore di test utilizza il seguente formato:

```
GroupId: group-id
CaseIds: [test-id, test-id] # optional
```

È necessario specificare `TestsoOneOfTests` per ogni funzione del `Features` elenco.

`OneOfTests`

Facoltativo. L'elenco dei descrittori di test. Almeno uno dei test specificati in questo elenco deve passare per supportare la funzionalità.

Ogni descrittore di test in questo elenco utilizza l'ID del gruppo di test e uno o più ID del test case per identificare i singoli test da eseguire da uno specifico gruppo di test. Il descrittore di test utilizza il seguente formato:

```
GroupId: group-id
CaseIds: [test-id, test-id] # optional
```

È necessario specificare `TestsoOneOfTests` per ogni funzione del `Features` elenco.

`IsRequired`

Il valore booleano che definisce se la feature è richiesta nel rapporto di test. Il valore di default è `false`.

Example

Testare il contesto dell'orchestrazione

Il contesto di test orchestrator è un documento JSON di sola lettura che contiene dati disponibili per l'orchestratore di test durante l'esecuzione. Il contesto dell'orchestratore di test è accessibile solo dall'orchestratore di test e contiene informazioni che determinano il flusso di test. Ad esempio, è possibile utilizzare le informazioni configurate dai test runner nel `userdata.json` file per determinare se è necessario eseguire un test specifico.

Il contesto dell'orchestratore di test utilizza il seguente formato:

```
{
    "pool": {
        <device-json-pool-element>
    },
    "userData": {
```

```
        <userdata-json-content>
    },
    "config": {
        <config-json-content>
    }
}
```

`pool`

Informazioni sul pool di dispositivi selezionato per l'esecuzione del test. Per un pool di dispositivi selezionato, queste informazioni vengono recuperate dall'elemento dell'array del pool di dispositivi di livello superiore corrispondente definito nel`device.json`file.

`userData`

Informazioni di nella`userdata.json`file.

`config`

Informazioni di nella`config.json`file.

È possibile eseguire una query sul contesto utilizzando la notazione JsonPath. La sintassi per le query JsonPath nelle definizioni di stato è`{{query}}`. Quando accedi ai dati dal contesto dell'orchestratore di test, assicurati che ogni valore venga valutato in una stringa, un numero o unbooleano.

Per ulteriori informazioni sull'uso della notazione JsonPath per l'accesso ai dati dal contesto, consultaUsa il contesto IDT.

Configurazione della macchina a stato IDT

> **Important**
>
> A partire da IDT v4.5.1, questa macchina a stato è obsoleta. Consigliamo vivamente di utilizzare il nuovo orchestratore di test. Per ulteriori informazioni, consulta la pagina Configurazione dell'orchestratore di test IDT .

Una macchina a stato è un costrutto che controlla il flusso di esecuzione della suite di test. Determina lo stato iniziale di una suite di test, gestisce le transizioni di stato in base a regole definite dall'utente e continua a passare attraverso tali stati fino a raggiungere lo stato finale.

Se la suite di test non include una macchina a stato definita dall'utente, IDT genererà una macchina a stato per te. La macchina a stato di default esegue le seguenti funzioni:

- Fornisce ai test runner la possibilità di selezionare ed eseguire gruppi di test specifici, anziché l'intera suite di test.
- Se non sono selezionati gruppi di test specifici, esegue tutti i gruppi di test nella suite di test in un ordine casuale.
- Genera report e stampa un riepilogo della console che mostra i risultati del test per ciascun gruppo di test e test case.

La macchina a stati per una suite di test IDT deve soddisfare i seguenti criteri:

- Ciascuno stato corrisponde a un'azione che IDT deve intraprendere, ad esempio per eseguire un gruppo di test o produrre un file di report.
- La transizione a uno stato esegue l'azione associata allo stato.
- Ciascuno stato definisce la regola di transizione per lo stato successivo.
- Lo stato finale deve essere`Succeed`o`Fail`.

Formato macchina a stati

È possibile utilizzare il seguente modello per configurare il proprio*<custom-test-suite-folder>*`/suite/state_machine.json`file:

```
{
  "Comment": "<description>",
  "StartAt": "<state-name>",
  "States": {
    "<state-name>": {
      "Type": "<state-type>",
      // Additional state configuration
    }

    // Required states
    "Succeed": {
      "Type": "Succeed"
    },
    "Fail": {
      "Type": "Fail"
```

```
    }
  }
}
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`Comment`

Una descrizione della macchina a stati.

`StartAt`

Il nome dello stato in cui IDT inizia a eseguire la suite di test. Il valore di`StartAt`deve essere impostato su uno degli stati elencati nella`States`oggetto.

`States`

Oggetto che mappa i nomi di stato definiti dall'utente a stati IDT validi. Ogni stato.*nome stato*object contiene la definizione di uno stato valido mappato al*nome stato*.

La`States`l'oggetto deve includere il`Succeede``Fail`stati. Per informazioni sugli stati validi, consultaStati e definizioni di stato validi.

Stati e definizioni di stato validi

Questa sezione descrive le definizioni di stato di tutti gli stati validi che possono essere utilizzati nella macchina a stato IDT. Alcuni dei seguenti stati supportano le configurazioni a livello di test case. Tuttavia, si consiglia di configurare le regole di transizione dello stato a livello di gruppo di test anziché a livello di test case, a meno che non sia assolutamente necessario.

Definizioni di stato

- RunTask
- Choice
- Parallel
- Aggiungi caratteristiche del prodotto
- Report
- Messaggio di log
- Seleziona gruppo
- Fail

- Succeed

RunTask

LaRunTaskLo stato esegue test cases da un gruppo di test definito nella suite di test.

```
{
    "Type": "RunTask",
    "Next": "<state-name>",
    "TestGroup": "<group-id>",
    "TestCases": [
        "<test-id>"
    ],
    "ResultVar": "<result-name>"
}
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`Next`

Il nome dello stato al quale passare dopo aver eseguito le azioni nello stato corrente.

`TestGroup`

Facoltativo. L'ID del gruppo di test da eseguire. Se questo valore non è specificato, IDT esegue il gruppo di test selezionato dal corridore di test.

`TestCases`

Facoltativo. Un array di ID del test case del gruppo specificato in`TestGroup`. In base ai valori di`TestGroup`e`TestCases`, IDT determina il comportamento di esecuzione del test come segue:

- Quando entrambi`TestGroup`e`TestCases`sono specificati, IDT esegue i test case specificati dal gruppo di test.
- Quando`TestCases`sono specificati ma`TestGroup`non è specificato, IDT esegue i test case specificati.
- Quando`TestGroup`è specificato, ma`TestCases`Non è specificato, IDT esegue tutti i test case all'interno del gruppo di test specificato.
- Quando nessuno dei due`TestGroup`o`TestCases`è specificato, IDT esegue tutti i test case del gruppo di test selezionato dal corridore di prova dalla CLI IDT. Per abilitare la selezione del gruppo per i test runner, è necessario includere entrambi`RunTask`e`Choice`stati

nel tuostate_machine.jsonfile. Per un esempio su come eseguire tale operazione, consultaEsempio della macchina a stati: Esegui gruppi di test selezionati dall'utente.

Per ulteriori informazioni sull'abilitazione di comandi CLI IDT per i test runner, consultathe section called “Abilita i comandi IDT CLI IDT”.

`ResultVar`

Il nome della variabile di contesto da impostare con i risultati dell'esecuzione del test. Non specificare questo valore se non hai specificato un valore per`TestGroup`. IDT imposta il valore della variabile definita in`ResultVar`a`true`o`false`in base a quanto segue:

- Se il nome della variabile è del modulo*`text`*`_`*`text`*`_passed`, quindi il valore viene impostato su se tutti i test del primo gruppo di test sono passati o sono stati saltati.
- In tutti gli altri casi, il valore è impostato su se tutti i test in tutti i gruppi di test sono stati superati o sono stati saltati.

In genere si utilizza`RunTask`stato per specificare un ID gruppo di test senza specificare gli ID del caso di test individuali, in modo che IDT esegua tutti i test case nel gruppo di test specificato. Tutti i test case gestiti da questo stato vengono eseguiti in parallel, in ordine casuale. Tuttavia, se tutti i test case richiedono l'esecuzione di un dispositivo ed è disponibile solo un singolo dispositivo, i test case verranno eseguiti in modo sequenziale.

Gestione errori

Se uno qualsiasi dei gruppi di test o ID del test case specificati non è valido, questo stato emette il`RunTaskError`Errore di esecuzione Se lo stato rileva un errore di esecuzione, imposta anche il`hasExecutionError`variabile nel contesto della macchina a stato per`true`.

Choice

La`Choice`state consente di impostare dinamicamente lo stato successivo in cui passare in base alle condizioni definite dall'utente.

```
{
    "Type": "Choice",
    "Default": "<state-name>",
    "FallthroughOnError": true | false,
    "Choices": [
        {
            "Expression": "<expression>",
            "Next": "<state-name>"
```

```
        }
    ]
}
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`Default`

Lo stato predefinito al quale passare se nessuna delle espressioni definite in`Choices`può essere valutato per`true`.

`FallthroughOnError`

Facoltativo. Specifica il comportamento quando lo stato rileva un errore nella valutazione delle espressioni. Impostare su`true`se si desidera saltare un'espressione se la valutazione genera un errore. Se nessuna espressione corrisponde, la macchina a stati passa alla`Default`Stato. Se il file`FallthroughOnError`Il valore non è specificato, il valore predefinito è`false`.

`Choices`

Una matrice di espressioni e stati per determinare a quale stato passare dopo aver eseguito le azioni nello stato corrente.

`Choices.Expression`

Una stringa di espressioni che restituisce un valore booleano. Se l'espressione viene valutata`true`, quindi la macchina a stati passa allo stato definito in`Choices.Next`. Le stringhe di espressione recuperano i valori dal contesto della macchina statale e quindi eseguono operazioni su di esse per arrivare a un valore booleano. Per informazioni sull'accesso al contesto della macchina a stato, vedereContesto della macchina a stati.

`Choices.Next`

Il nome dello stato al quale passare se l'espressione definita in`Choices.Expression`valuta su`true`.

Gestione errori

La`Choices`stato può richiedere la gestione degli errori nei seguenti casi:

- Alcune variabili nelle espressioni di scelta non esistono nel contesto della macchina a stato.
- Il risultato di un'espressione non è un valore booleano.
- Il risultato di una ricerca JSON non è una stringa, un numero o un valore booleano.

Non è possibile utilizzare un`Catch`Blocca per gestire gli errori in questo stato. Se si desidera interrompere l'esecuzione della macchina a stato quando viene rilevato un errore, è necessario impostare`FallthroughOnError`a`false`. Consigliamo, tuttavia, di impostare`FallthroughOnError`a`true`e in base al caso d'uso, esegui una delle seguenti operazioni:

- Se in alcuni casi una variabile a cui si sta accedendo non esiste, utilizzare il valore di`Default`e addizionale`Choices`blocchi per specificare lo stato successivo.
- Se una variabile a cui si accede deve sempre esistere, impostare il`Default`a stati`Fail`.

Parallel

La`Parallel`state consente di definire ed eseguire nuove macchine a stato parallel l'una con l'altra.

```
{
    "Type": "Parallel",
    "Next": "<state-name>",
    "Branches": [
        <state-machine-definition>
    ]
}
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`Next`

Il nome dello stato al quale passare dopo aver eseguito le azioni nello stato corrente.

`Branches`

Un array di definizioni delle macchine statali da eseguire. Ogni definizione della macchina a stati deve contenere una propria`StartAt`,`Succeed`, e`Fail`stati. Le definizioni della macchina di stato in questo array non possono fare riferimento a stati al di fuori della propria definizione.

> **Note**
>
> Poiché ogni macchina a stato di ramo condivide lo stesso contesto macchina a stato, l'impostazione di variabili in un ramo e quindi la lettura di tali variabili da un altro ramo potrebbe comportare un comportamento imprevisto.

La`Parallel`stato si sposta allo stato successivo solo dopo aver eseguito tutte le macchine dello stato del ramo. Ogni stato che richiede un dispositivo aspetterà l'esecuzione fino a quando il dispositivo non sarà disponibile. Se sono disponibili più dispositivi, questo stato esegue i test case di più gruppi in parallel. Se non sono disponibili dispositivi sufficienti, i test case verranno eseguiti in sequenza. Poiché i test case vengono eseguiti in ordine casuale quando vengono eseguiti in parallel, è possibile utilizzare dispositivi diversi per eseguire test dello stesso gruppo di test.

Gestione errori

Assicurarsi che sia la macchina a stato di diramazione che la macchina a stato padre passino al`Fail`stato per gestire gli errori di esecuzione.

Poiché le macchine a stato di diramazione non trasmettono errori di esecuzione alla macchina a stato padre, non è possibile utilizzare un`Catch`blocco per gestire gli errori di esecuzione nelle macchine a stato di filiale. Utilizza invece`hasExecutionErrors`valore nel contesto della macchina a stato condiviso. Per un esempio su come eseguire tale operazione, consultaEsempio della macchina a stati: Esegui due gruppi di test in parallel.

Aggiungi caratteristiche del prodotto

La`AddProductFeatures`stato consente di aggiungere funzionalità del prodotto a`awsiotdevicetester_report.xml`file generato da IDT.

Una funzione di prodotto è costituita da informazioni definite dall'utente su criteri specifici che un dispositivo potrebbe soddisfare. Ad esempio, le ricette`MQTT`la funzionalità del prodotto può indicare che il dispositivo pubblica correttamente i messaggi MQTT. Nel report, le funzionalità del prodotto sono impostate come`supported`,`not-supported`o un valore personalizzato, in base al superamento dei test specificati.

> **Note**
>
> La`AddProductFeatures`stato non genera rapporti da solo. Questo stato deve passare al`Reportstate`per generare report.

```
{
    "Type": "Parallel",
    "Next": "<state-name>",
    "Features": [
```

```
        {
            "Feature": "<feature-name>",
            "Groups": [
                "<group-id>"
            ],
            "OneOfGroups": [
                "<group-id>"
            ],
            "TestCases": [
                "<test-id>"
            ],
            "IsRequired": true | false,
            "ExecutionMethods": [
                "<execution-method>"
            ]
        }
    ]
}
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`Next`

Il nome dello stato al quale passare dopo aver eseguito le azioni nello stato corrente.

`Features`

Una serie di funzionalità del prodotto da mostrare nella`awsiotdevicetester_report.xml`file.

`Feature`

Il nome della caratteristica

`FeatureValue`

Facoltativo. Il valore personalizzato da utilizzare nel report anzichè`supported`. Se questo valore non è specificato, in base ai risultati del test, il valore della feature viene impostato su`supported`o`not-supported`.

Se si utilizza un valore personalizzato per`FeatureValue`, è possibile testare la stessa funzionalità con condizioni diverse e IDT concatena i valori delle feature per le condizioni supportate. Ad esempio, il seguente estratto mostra il`MyFeature`feature con due valori di feature separati:

```
...
```

```
{
    "Feature": "MyFeature",
    "FeatureValue": "first-feature-supported",
    "Groups": ["first-feature-group"]
},
{
    "Feature": "MyFeature",
    "FeatureValue": "second-feature-supported",
    "Groups": ["second-feature-group"]
},
...
```

Se entrambi i gruppi di test passano, il valore della feature è impostato su`first-feature-supported, second-feature-supported`.

`Groups`

Facoltativo. Un array di ID del gruppo di test. Tutti i test all'interno di ciascun gruppo di test specificato devono essere superati per supportare la funzionalità.

`OneOfGroups`

Facoltativo. Un array di ID del gruppo di test. Tutti i test all'interno di almeno uno dei gruppi di test specificati devono essere superati affinché la funzionalità sia supportata.

`TestCases`

Facoltativo. Un array di ID del test case. Se si specifica questo valore, si applica quanto segue:

- Tutti i test case specificati devono essere passati per supportare la funzionalità.
- `Groups`deve contenere un solo ID del gruppo di test.
- `OneOfGroups`non deve essere specificato.

`IsRequired`

Facoltativo. Impostare su`false`per contrassegnare questa funzione come funzione facoltativa nel report. Il valore di default è `true`.

`ExecutionMethods`

Facoltativo. Un array di metodi di esecuzione che corrispondono al`protocol`valore specificato nel`device.json`file. Se questo valore è specificato, i test runner devono specificare un`protocol`valore che corrisponde a uno dei valori di questa matrice per includere la feature nel report. Se questo valore non viene specificato, la funzione verrà sempre inclusa nel report.

Per utilizzare il pluginAddProductFeaturesstato, è necessario impostare il valore diResultVarnellaRunTaskstato su uno dei valori seguenti:

- Se sono stati specificati i singoli ID del test case, impostareResultVara*group-id_test-id*_passed.
- Se non sono stati specificati singoli ID test case, impostareResultVara*group-id*_passed.

LaAddProductFeaturesVerifica dello stato dei risultati dei test nel modo seguente:

- Se non è stato specificato alcun ID del test case, il risultato per ciascun gruppo di test viene determinato dal valore del*group-id*_passedvariabile nel contesto della macchina a stato.
- Se hai specificato gli ID del test case, il risultato per ciascuno dei test viene determinato dal valore del*group-id_test-id*_passedvariabile nel contesto della macchina a stato.

Gestione errori

Se un ID di gruppo fornito in questo stato non è un ID di gruppo valido, questo stato si traduce nellaAddProductFeaturesErrorErrore di esecuzione Se lo stato rileva un errore di esecuzione, imposta anche ilhasExecutionErrorsvariabile nel contesto della macchina a stato pertrue.

Report

LaReportgenera il*suite-name*_Report.xmleawsiotdevicetester_report.xmlfile. Questo stato trasmette anche il report sulla console.

```
{
    "Type": "Report",
    "Next": "<state-name>"
}
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`Next`

Il nome dello stato al quale passare dopo aver eseguito le azioni nello stato corrente.

Dovresti sempre passare alReportstato verso la fine del flusso di esecuzione del test in modo che i test runner possano visualizzare i risultati del test. In genere, lo stato successivo dopo questo stato èSucceed.

Gestione errori

Se questo stato incontra problemi con la generazione dei report, emette il`ReportError`Errore di esecuzione

Messaggio di log

La`LogMessage`genera il`test_manager.log`File e trasmette il messaggio di log nella console.

```
{
    "Type": "LogMessage",
    "Next": "<state-name>"
    "Level": "info | warn | error"
    "Message": "<message>"
}
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`Next`

Il nome dello stato al quale passare dopo aver eseguito le azioni nello stato corrente.

`Level`

Il livello di errore al quale creare il messaggio di registro. Se si specifica un livello non valido, questo stato genera un messaggio di errore e lo scarta.

`Message`

Il messaggio da registrare.

Seleziona gruppo

La`SelectGroup`state aggiorna il contesto della macchina statale per indicare quali gruppi sono selezionati. I valori impostati da questo stato vengono utilizzati da qualsiasi successivo`Choice`stati.

```
{
    "Type": "SelectGroup",
    "Next": "<state-name>"
    "TestGroups": [
        <group-id>"
    ]
```

```
}
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`Next`

Il nome dello stato al quale passare dopo aver eseguito le azioni nello stato corrente.

`TestGroups`

Una serie di gruppi di test che verranno contrassegnati come selezionati. Per ogni ID gruppo di test in questo array, il*`group-id`*`_selected`variabile è impostata su`true`nel contesto. Assicurati di fornire ID del gruppo di test validi perché IDT non convalida se esistono i gruppi specificati.

Fail

La`Fail`stato indica che la macchina a stato non è stata eseguita correttamente. Si tratta di uno stato finale per la macchina a stato e ogni definizione della macchina a stato deve includere questo stato.

```
{
    "Type": "Fail"
}
```

Succeed

La`Succeed`stato indica che la macchina a stato è stata eseguita correttamente. Si tratta di uno stato finale per la macchina a stato e ogni definizione della macchina a stato deve includere questo stato.

```
{
    "Type": "Succeed"
}
```

Contesto della macchina a stati

Il contesto macchina a stato è un documento JSON di sola lettura che contiene dati disponibili per la macchina a stato durante l'esecuzione. Il contesto della macchina a stato è accessibile solo dalla macchina a stato e contiene informazioni che determinano il flusso di test. Ad esempio, è possibile utilizzare le informazioni configurate dai test runner nel`userdata.json`file per determinare se è necessario eseguire un test specifico.

Il contesto della macchina a stati utilizza il formato seguente:

```
{
    "pool": {
        <device-json-pool-element>
    },
    "userData": {
        <userdata-json-content>
    },
    "config": {
        <config-json-content>
    },
    "suiteFailed": true | false,
    "specificTestGroups": [
        "<group-id>"
    ],
    "specificTestCases": [
        "<test-id>"
    ],
    "hasExecutionErrors": true
}
```

`pool`

Informazioni sul pool di dispositivi selezionato per l'esecuzione del test. Per un pool di dispositivi selezionato, queste informazioni vengono recuperate dall'elemento dell'array del pool di dispositivi di livello superiore corrispondente definito nel`device.json`file.

`userData`

Informazioni nel`userdata.json`file.

`config`

Informazioni di pin`config.json`file.

`suiteFailed`

Il valore è impostato su`false`Quando si avvia la macchina a stati. Se un gruppo di test fallisce in un`RunTask`stato, quindi questo valore è impostato su`true`per la durata rimanente dell'esecuzione della macchina a stati.

`specificTestGroups`

Se il test runner seleziona gruppi di test specifici da eseguire al posto dell'intera suite di test, questa chiave viene creata e contiene l'elenco di ID specifici del gruppo di test.

`specificTestCases`

Se il test runner seleziona casi di test specifici da eseguire al posto dell'intera suite di test, questa chiave viene creata e contiene l'elenco degli ID del test case specifici.

`hasExecutionErrors`

Non esce quando si avvia la macchina a stato. Se uno stato incontra errori di esecuzione, questa variabile viene creata e impostata su`true`per la durata rimanente dell'esecuzione della macchina a stati.

È possibile eseguire una query sul contesto utilizzando la notazione JsonPath. La sintassi per le query JsonPath nelle definizioni di stato è`{{$.query}}`. È possibile utilizzare le query JsonPath come stringhe segnaposto all'interno di alcuni stati. IDT sostituisce le stringhe segnaposto con il valore della query JsonPath valutata dal contesto. È possibile utilizzare segnaposto per i seguenti valori:

- La`TestCases`valore in`RunTask`stati.
- La`Expression`valore`Choice`Stato.

Quando accedi ai dati dal contesto della macchina a stati, assicurati che siano soddisfatte le seguenti condizioni:

- I percorsi JSON devono iniziare con`$.`
- Ogni valore deve essere valutato in una stringa, un numero o un valore booleano.

Per ulteriori informazioni sull'utilizzo della notazione JsonPath per accedere ai dati dal contesto, consultaUsa il contesto IDT.

Errori di esecuzione

Gli errori di esecuzione sono errori nella definizione della macchina a stato rilevata dalla macchina a stato durante l'esecuzione di uno stato. IDT registra le informazioni su ogni errore nel`test_manager.log`File e trasmette il messaggio di log nella console.

È possibile utilizzare i seguenti metodi per gestire gli errori di esecuzione:

- Aggiungi un`Catch`bloccarenella definizione dello stato.
- Verificare il valore di`hasExecutionErrors`valorenel contesto della macchina a stati.

Cattura

Per utilizzare`Catch`, aggiungi quanto segue alla tua definizione di stato:

```
"Catch": [
    {
        "ErrorEquals": [
            "<error-type>"
        ]
        "Next": "<state-name>"
    }
]
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`Catch.ErrorEquals`

Un array dei tipi di errori da catch. Se un errore di esecuzione corrisponde a uno dei valori specificati, la macchina a stati passa allo stato specificato in`Catch.Next`. Vedere ciascuna definizione di stato per informazioni sul tipo di errore che produce.

`Catch.Next`

Lo stato successivo a cui passare se lo stato corrente rileva un errore di esecuzione corrispondente a uno dei valori specificati in`Catch.ErrorEquals`.

I blocchi di cattura vengono gestiti in sequenza fino a quando uno corrisponde. Se gli errori non corrispondono a quelli elencati nei blocchi Catch, le macchine a stato continuano ad essere eseguite. Poiché gli errori di esecuzione sono il risultato di definizioni di stato errate, si consiglia di passare allo stato Fail quando uno stato incontra un errore di esecuzione.

Errore di esecuzione

Quando alcuni stati incontrano errori di esecuzione, oltre a emettere l'errore, impostano anche il`hasExecutionError`valore `atrue`nel contesto della macchina a stati. È possibile utilizzare questo valore per rilevare quando si verifica un errore e quindi utilizzare un`Choice`stato per la transizione della macchina statale al`Fail`Stato.

Questo metodo presenta le seguenti caratteristiche.

- La macchina a stato non inizia con alcun valore assegnato`hasExecutionError`e questo valore non è disponibile fino a quando uno stato particolare non lo imposta. Ciò significa che è necessario

impostare esplicitamente il`FallthroughOnError`a`false`per`Choice`afferma che accede a questo valore per impedire l'arresto della macchina a stato se non si verificano errori di esecuzione.

- Una volta impostato su`true`,`hasExecutionError`non è mai impostato su false o rimosso dal contesto. Ciò significa che questo valore è utile solo la prima volta che viene impostato su`true`e per tutti gli stati successivi, non fornisce un valore significativo.
- La`hasExecutionError`il valore è condiviso con tutte le macchine a stato di diramazione`Parallel`stato, che può portare a risultati imprevisti a seconda dell'ordine in cui si accede.

A causa di queste caratteristiche, non è consigliabile utilizzare questo metodo se è possibile utilizzare invece un blocco Catch.

Esempio delle macchine a stati

Questa sezione fornisce alcuni esempi di configurazioni della macchina a stati.

Esempi

- Esempio della macchina a stati: Esegui un singolo gruppo di test
- Esempio della macchina a stati: Esegui gruppi di test selezionati dall'utente
- Esempio della macchina a stati: Esegui un singolo gruppo di test con caratteristiche del prodotto
- Esempio della macchina a stati: Esegui due gruppi di test in parallel

Esempio della macchina a stati: Esegui un singolo gruppo di test

Questa macchina a stati:

- Esegue il gruppo di test con id`GroupA`, che deve essere presente nella suite in un`group.json`file.
- Verifica la presenza di errori di esecuzione e transizioni a`Fail`se ne vengono trovati.
- Genera un report e transizioni a`Succeed`se non ci sono errori e`Fail`In caso contrario, .

```
{
    "Comment": "Runs a single group and then generates a report.",
    "StartAt": "RunGroupA",
    "States": {
        "RunGroupA": {
            "Type": "RunTask",
```

```
            "Next": "Report",
            "TestGroup": "GroupA",
            "Catch": [
                {
                    "ErrorEquals": [
                        "RunTaskError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Report": {
            "Type": "Report",
            "Next": "Succeed",
            "Catch": [
                {
                    "ErrorEquals": [
                        "ReportError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Succeed": {
            "Type": "Succeed"
        },
        "Fail": {
            "Type": "Fail"
        }
    }
}
```

Esempio della macchina a stati: Esegui gruppi di test selezionati dall'utente

Questa macchina a stati:

- Verifica se il corridore di prova ha selezionato gruppi di test specifici. La macchina a stato non verifica la presenza di casi di prova specifici perché i test runner non possono selezionare i test case senza selezionare anche un gruppo di test.
- Se sono selezionati gruppi di test:
  - Esegue i test case all'interno dei gruppi di test selezionati. A tale scopo, la macchina a stato non specifica esplicitamente alcun gruppo di test o test case nel`RunTask`Stato.

  - Genera un report dopo aver eseguito tutti i test e le uscite.
- Se i gruppi di test non sono selezionati:
  - Esegue test nel gruppo di test`GroupA`.
  - Genera report ed uscite.

```
{
    "Comment": "Runs specific groups if the test runner chose to do that, otherwise
 runs GroupA.",
    "StartAt": "SpecificGroupsCheck",
    "States": {
        "SpecificGroupsCheck": {
            "Type": "Choice",
            "Default": "RunGroupA",
            "FallthroughOnError": true,
            "Choices": [
                {
                    "Expression": "{{$.specificTestGroups[0]}} != ''",
                    "Next": "RunSpecificGroups"
                }
            ]
        },
        "RunSpecificGroups": {
            "Type": "RunTask",
            "Next": "Report",
            "Catch": [
                {
                    "ErrorEquals": [
                        "RunTaskError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "RunGroupA": {
            "Type": "RunTask",
            "Next": "Report",
            "TestGroup": "GroupA",
            "Catch": [
                {
                    "ErrorEquals": [
                        "RunTaskError"
```

```
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Report": {
            "Type": "Report",
            "Next": "Succeed",
            "Catch": [
                {
                    "ErrorEquals": [
                        "ReportError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Succeed": {
            "Type": "Succeed"
        },
        "Fail": {
            "Type": "Fail"
        }
    }
}
```

Esempio della macchina a stati: Esegui un singolo gruppo di test con caratteristiche del prodotto

Questa macchina a stati:

- Eseguire il gruppo di test`GroupA`.
- Verifica la presenza di errori di esecuzione e transizioni a`Fail`se ne vengono trovati.
- Aggiunge`FeatureThatDependsOnGroupA`caratteristica dela`wsiotdevicetester_report.xml`file:
  - Se`GroupA`passa, la caratteristica è impostata su`supported`.
  - La funzione non è contrassegnata come facoltativa nel report.
- Genera un report e transizioni a`Succeed`se non ci sono errori e`Fail`altrimenti

```
{
    "Comment": "Runs GroupA and adds product features based on GroupA",
```

```
    "StartAt": "RunGroupA",
    "States": {
        "RunGroupA": {
            "Type": "RunTask",
            "Next": "AddProductFeatures",
            "TestGroup": "GroupA",
            "ResultVar": "GroupA_passed",
            "Catch": [
                {
                    "ErrorEquals": [
                        "RunTaskError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "AddProductFeatures": {
            "Type": "AddProductFeatures",
            "Next": "Report",
            "Features": [
                {
                    "Feature": "FeatureThatDependsOnGroupA",
                    "Groups": [
                        "GroupA"
                    ],
                    "IsRequired": true
                }
            ]
        },
        "Report": {
            "Type": "Report",
            "Next": "Succeed",
            "Catch": [
                {
                    "ErrorEquals": [
                        "ReportError"
                    ],
                    "Next": "Fail"
                }
            ]
        },
        "Succeed": {
            "Type": "Succeed"
        },
```

```
        "Fail": {
            "Type": "Fail"
        }
    }
}
```

Esempio della macchina a stati: Esegui due gruppi di test in parallel

Questa macchina a stati:

- Eseguire`GroupA`e`GroupB`gruppi di test in parallel. La`ResultVar`variabili memorizzate nel contesto dal`RunTask`gli stati nelle macchine dello stato di filiale sono disponibili per il`AddProductFeatures`Stato.
- Verifica la presenza di errori di esecuzione e transizioni a`Fail`se ne vengono trovati. Questa macchina a stato non utilizza un`Catch`blocco perché tale metodo non rileva errori di esecuzione nelle macchine a stato di diramazione.
- Aggiunge funzionalità al`awsiotdevicetester_report.xml`file basato sui gruppi che passano
  - Se`GroupA`passa, la funzione è impostata su`supported`.
  - La funzione non è contrassegnata come facoltativa nel report.
- Genera un report e transizioni a`Succeed`se non ci sono errori e`Fail`altrimenti

Se nel pool di dispositivi sono configurati due dispositivi, entrambi`GroupA`e`GroupB`può eseguire contemporaneamente. Tuttavia, se uno dei due`GroupA`o`GroupB`contiene più test, quindi entrambi i dispositivi possono essere assegnati a tali test. Se è configurato un solo dispositivo, i gruppi di test verranno eseguiti in sequenza.

```
{
    "Comment": "Runs GroupA and GroupB in parallel",
    "StartAt": "RunGroupAAndB",
    "States": {
        "RunGroupAAndB": {
            "Type": "Parallel",
            "Next": "CheckForErrors",
            "Branches": [
                {
                    "Comment": "Run GroupA state machine",
                    "StartAt": "RunGroupA",
                    "States": {
                        "RunGroupA": {
```

```
                    "Type": "RunTask",
                    "Next": "Succeed",
                    "TestGroup": "GroupA",
                    "ResultVar": "GroupA_passed",
                    "Catch": [
                        {
                            "ErrorEquals": [
                                "RunTaskError"
                            ],
                            "Next": "Fail"
                        }
                    ]
                },
                "Succeed": {
                    "Type": "Succeed"
                },
                "Fail": {
                    "Type": "Fail"
                }
            }
        },
        {
            "Comment": "Run GroupB state machine",
            "StartAt": "RunGroupB",
            "States": {
                "RunGroupA": {
                    "Type": "RunTask",
                    "Next": "Succeed",
                    "TestGroup": "GroupB",
                    "ResultVar": "GroupB_passed",
                    "Catch": [
                        {
                            "ErrorEquals": [
                                "RunTaskError"
                            ],
                            "Next": "Fail"
                        }
                    ]
                },
                "Succeed": {
                    "Type": "Succeed"
                },
                "Fail": {
                    "Type": "Fail"
```

```
                    }
                }
            }
        ]
    },
    "CheckForErrors": {
        "Type": "Choice",
        "Default": "AddProductFeatures",
        "FallthroughOnError": true,
        "Choices": [
            {
                "Expression": "{{$.hasExecutionErrors}} == true",
                "Next": "Fail"
            }
        ]
    },
    "AddProductFeatures": {
        "Type": "AddProductFeatures",
        "Next": "Report",
        "Features": [
            {
                "Feature": "FeatureThatDependsOnGroupA",
                "Groups": [
                    "GroupA"
                ],
                "IsRequired": true
            },
            {
                "Feature": "FeatureThatDependsOnGroupB",
                "Groups": [
                    "GroupB"
                ],
                "IsRequired": true
            }
        ]
    },
    "Report": {
        "Type": "Report",
        "Next": "Succeed",
        "Catch": [
            {
                "ErrorEquals": [
                    "ReportError"
                ],
```

```
                    "Next": "Fail"
                }
            ]
        },
        "Succeed": {
            "Type": "Succeed"
        },
        "Fail": {
            "Type": "Fail"
        }
    }
}
```

Crea file eseguibili per test case IDT

È possibile creare e inserire gli eseguibili dei test case in una cartella della suite di test nei seguenti modi:

- Per le suite di test che utilizzano argomenti o variabili di ambiente dai`test.json` file per determinare quali test eseguire, è possibile creare un singolo test case eseguibile per l'intera suite di test o un eseguibile di test per ogni gruppo di test nella suite di test.
- Per una suite di test in cui si desidera eseguire test specifici in base a comandi specificati, si crea un test case eseguibile per ogni test case nella suite di test.

In qualità di autore di test, puoi determinare quale approccio è appropriato per il tuo caso d'uso e strutturare di conseguenza il tuo test case eseguibile. Assicurati di fornire il percorso eseguibile del test case corretto in ogni`test.json` file e che l'eseguibile specificato venga eseguito correttamente.

Quando tutti i dispositivi sono pronti per l'esecuzione di un test case, IDT legge i seguenti file:

- Il test case`test.json` per il test selezionato determina i processi da avviare e le variabili di ambiente da impostare.
- La suite`suite.json` for the test determina le variabili di ambiente da impostare.

IDT avvia il processo eseguibile di test richiesto in base ai comandi e agli argomenti specificati nel`test.json` file e passa le variabili di ambiente richieste al processo.

Usa l'IDT Client SDK

Gli IDT Client SDK consentono di semplificare il modo in cui si scrive la logica di test nell'eseguibile di test con comandi API che è possibile utilizzare per interagire con IDT e i dispositivi in fase di test. IDT attualmente fornisce i seguenti SDK:

- SDK per Python
- SDK for Go di DAF per Go
- SDK for Java

Questi SDK si trovano nella*<device-tester-extract-location>*`/sdks` cartella. Quando si crea un nuovo eseguibile del test case, è necessario copiare l'SDK che si desidera utilizzare nella cartella contenente l'eseguibile del test case e fare riferimento all'SDK nel codice. Questa sezione fornisce una breve descrizione dei comandi API disponibili che puoi utilizzare negli eseguibili del test case.

In questa sezione

Interazione dei dispositivi

I seguenti comandi consentono di comunicare con il dispositivo in prova senza dover implementare funzioni aggiuntive di interazione del dispositivo e gestione della connettività.

`ExecuteOnDevice`

Consente alle suite di test di eseguire comandi shell su un dispositivo che supporta connessioni shell SSH o Docker.

`CopyToDevice`

Consente alle suite di test di copiare un file locale dalla macchina host che esegue IDT in una posizione specificata su un dispositivo che supporta connessioni shell SSH o Docker.

`ReadFromDevice`

Consente alle suite di test di leggere dalla porta seriale dei dispositivi che supportano le connessioni UART.

> **Note**
>
> Poiché IDT non gestisce le connessioni dirette ai dispositivi realizzate utilizzando le informazioni di accesso ai dispositivi dal contesto, consigliamo di utilizzare questi comandi API di interazione con i dispositivi negli eseguibili del test case. Tuttavia, se questi comandi non soddisfano i requisiti del test case, puoi recuperare le informazioni di accesso al dispositivo dal contesto IDT e utilizzarle per stabilire una connessione diretta al dispositivo dalla suite di test.
> Per stabilire una connessione diretta, recupera le informazioni nei campi`device.connectivity` `eresource.devices.connectivity` nei campi rispettivamente per il dispositivo sottoposto a test e per i dispositivi di risorse. Per ulteriori informazioni sull'utilizzo del contesto IDT, consultaUsa il contesto IDT.

Interazione IDT

I seguenti comandi consentono alle suite di test di comunicare con IDT.

`PollForNotifications`

Consente alle suite di test di verificare la presenza di notifiche da IDT.

`GetContextValue` e `GetContextString`

Consente alle suite di test di recuperare valori dal contesto IDT. Per ulteriori informazioni, consulta Usa il contesto IDT.

`SendResult`

Consente alle suite di test di segnalare i risultati dei test case all'IDT. Questo comando deve essere chiamato alla fine di ogni test case in una suite di test.

Interazione dell'host

Il comando seguente consente alle suite di test di comunicare con la macchina host.

`PollForNotifications`

Consente alle suite di test di verificare la presenza di notifiche da IDT.

`GetContextValue` e `GetContextString`

Consente alle suite di test di recuperare valori dal contesto IDT. Per ulteriori informazioni, consulta Usa il contesto IDT.

`ExecuteOnHost`

Consente alle suite di test di eseguire comandi sul computer locale e consente a IDT di gestire il ciclo di vita dell'eseguibile del test case.

Abilita i comandi IDT CLI IDT

Il `run-suite` comando IDT CLI fornisce diverse opzioni che consentono al test runner di personalizzare l'esecuzione del test. Per consentire ai test runner di utilizzare queste opzioni per eseguire la tua suite di test personalizzata, implementi il supporto per l'IDT CLI. Se non si implementa il supporto, i test runner saranno comunque in grado di eseguire i test, ma alcune opzioni della CLI non funzioneranno correttamente. Per offrire un'esperienza cliente ideale, si consiglia di implementare il supporto per i seguenti argomenti del `run-suite` comando nella CLI IDT:

`timeout-multiplier`

Specifica un valore maggiore di 1,0 che verrà applicato a tutti i timeout durante l'esecuzione dei test.

I test runner possono utilizzare questo argomento per aumentare il timeout per i casi di test che desiderano eseguire. Quando un test runner specifica questo argomento nel proprio `run-suite` comando, IDT lo utilizza per calcolare il valore della variabile di ambiente IDT_TEST_TIMEOUT e imposta il `config.timeoutMultiplier` campo nel contesto IDT. Per supportare questa argomentazione, è necessario eseguire le operazioni indicate di seguito:

- Invece di utilizzare direttamente il valore di timeout dal `test.json` file, leggi la variabile di ambiente IDT_TEST_TIMEOUT per ottenere il valore di timeout calcolato correttamente.
- Recupera il `config.timeoutMultiplier` valore dal contesto IDT e applicalo a timeout di lunga durata.

Per ulteriori informazioni sull'uscita anticipata a causa di eventi di timeout, consulta Specifica il comportamento di uscita.

`stop-on-first-failure`

Specifica che IDT deve interrompere l'esecuzione di tutti i test in caso di errore.

Quando un test runner specifica questo argomento nel proprio`run-suite` comando, IDT interromperà l'esecuzione dei test non appena riscontra un errore. Tuttavia, se i casi di test vengono eseguiti in parallel, ciò può portare a risultati imprevisti. Per implementare il supporto, assicurati che se IDT rileva questo evento, la logica del test indichi a tutti i casi di test in esecuzione di interrompersi, ripulire le risorse temporanee e segnalare un risultato del test a IDT. Per ulteriori informazioni sull'errori dei processi, consultaSpecifica il comportamento di uscita.

`group-id` e `test-id`

Specifica che IDT deve eseguire solo i gruppi di test o i casi di test selezionati.

I test runner possono utilizzare questi argomenti con il loro`run-suite` comando per specificare il seguente comportamento di esecuzione del test:

- Esegui tutti i test all'interno dei gruppi di test specificati.
- Esegui una selezione di test all'interno di un gruppo di test specificato.

Per supportare questi argomenti, l'orchestratore di test per la tua suite di test deve includere un set specifico di`RunTask` e`Choice` stati nel tuo orchestratore di test. Se non si utilizza una macchina a stati personalizzata, l'orchestratore di test IDT predefinito include automaticamente gli stati richiesti e non è necessario eseguire ulteriori azioni. Tuttavia, se utilizzi un orchestratore di test personalizzato, utilizzaloEsempio della macchina a stati: Esegui gruppi di test selezionati dall'utente come campione per aggiungere gli stati richiesti nel tuo orchestratore di test.

Per ulteriori informazioni sui comandi CLI IDT, consultaEsegui il debug ed esegui suite di test personalizzate.

Scrivere registri degli eventi

Mentre il test è in esecuzione, si inviano dati`stdout` e si scrivono`stderr` i registri degli eventi e i messaggi di errore alla console. Per informazioni sul formato di messaggi della console, consultaFormato dei messaggi della console.

Quando l'IDT termina l'esecuzione della suite di test, queste informazioni sono disponibili anche nel`test_manager.log` file che si trova nella*`<devicetester-extract-location>`*`/results/`*`<execution-id>`*`/logs` cartella.

È possibile configurare ogni test case per scrivere i registri dell'esecuzione del test, inclusi i registri del dispositivo sottoposto a test, nel*`<group-id>`*`_`*`<test-id>`* file che si trova nella*`<device-tester-extract-location>`*`/results/`*`execution-id`*`/logs` cartella. A tale scopo, recupera il percorso del file di registro dal contesto IDT con `latestData.logFilePath` query, crea un file in

quel percorso e scrivi il contenuto che desideri. IDT aggiorna automaticamente il percorso in base al test case in esecuzione. Se scegli di non creare il file di registro per un test case, non viene generato alcun file per quel test case.

Puoi anche configurare il tuo eseguibile di testo per creare file di registro aggiuntivi, se necessario, nella*<device-tester-extract-location>*`/logs` cartella. Si consiglia di specificare prefissi univoci per i nomi dei file di registro in modo che i file non vengano sovrascritti.

Segnala i risultati a IDT

IDT scrive i risultati dei test nei file`awsiotdevicetester_report.xml` e nei*suite-name*`_report.xml` file. Questi file di report si trovano in*<device-tester-extract-location>*`/results/`*<execution-id>*`/`. Entrambi i report acquisiscono i risultati dell'esecuzione della suite di test. Per ulteriori informazioni sugli schemi utilizzati da IDT per questi report, vedereRivedi i risultati e i registri dei test

Per compilare il contenuto del*suite-name*`_report.xml` file, è necessario utilizzare il`SendResult` comando per segnalare i risultati del test a IDT prima del termine dell'esecuzione del test. Se IDT non è in grado di individuare i risultati di un test, emette un errore per il test case. Il seguente estratto di Python mostra i comandi per inviare un risultato del test a IDT:

```
request-variable = SendResultRequest(TestResult(result))
client.send_result(request-variable)
```

Se non si segnalano i risultati tramite l'API, IDT cerca i risultati dei test nella cartella degli artefatti del test. Il percorso di questa cartella è`testData.testArtifactsPath` memorizzato nel campo nel contesto IDT. In questa cartella, IDT utilizza il primo file XML in ordine alfabetico che trova come risultato del test.

Se la logica del test produce risultati XML JUnit, puoi scrivere i risultati del test in un file XML nella cartella degli artefatti per fornire direttamente i risultati a IDT invece di analizzarli e quindi utilizzare l'API per inviarli a IDT.

Se utilizzi questo metodo, assicurati che la logica del test riassuma accuratamente i risultati del test e formatti il file dei risultati nello stesso formato del*suite-name*`_report.xml` file. IDT non esegue alcuna convalida dei dati forniti, con le seguenti eccezioni:

- IDT ignora tutte le proprietà del`testsuites` tag. Invece, calcola le proprietà dei tag in base ai risultati di altri gruppi di test riportati.
- Al suo interno deve essere presente almeno un`testsuite` tag`testsuites`.

Poiché IDT utilizza la stessa cartella degli artefatti per tutti i casi di test e non elimina i file dei risultati tra le esecuzioni del test, questo metodo potrebbe anche portare a segnalazioni errate se IDT legge il file errato. Ti consigliamo di utilizzare lo stesso nome per il file dei risultati XML generato in tutti i casi di test per sovrascrivere i risultati di ogni test case e assicurarti che i risultati corretti siano disponibili per l'uso da IDT. Sebbene sia possibile utilizzare un approccio misto alla creazione di report nella suite di test, ovvero utilizzare un file di risultati XML per alcuni casi di test e inviare i risultati tramite l'API per altri, non consigliamo questo approccio.

Specifica il comportamento di uscita

Configura il tuo eseguibile di testo in modo che esca sempre con un codice di uscita pari a 0, anche se un test case riporta un errore o un risultato di errore. Utilizza codici di uscita diversi da zero solo per indicare che un test case non è stato eseguito o se l'eseguibile del test case non è stato in grado di comunicare alcun risultato a IDT. Quando IDT riceve un codice di uscita diverso da zero, indica che il test case ha riscontrato un errore che ne ha impedito l'esecuzione.

IDT potrebbe richiedere o aspettarsi che un test case smetta di funzionare prima che sia terminato nei seguenti eventi. Utilizza queste informazioni per configurare l'eseguibile del test case per rilevare ciascuno di questi eventi dal test case:

Timeout

Si verifica quando un test case viene eseguito per un periodo superiore al valore di timeout specificato nel`test.json` file. Se il test runner ha utilizzato l'`timeout-multiplier`argomento per specificare un moltiplicatore di timeout, IDT calcola il valore di timeout con il moltiplicatore.

Per rilevare questo evento, utilizzare la variabile di ambiente IDT_TEST_TIMEOUT. Quando un test runner avvia un test, IDT imposta il valore della variabile di ambiente IDT_TEST_TIMEOUT sul valore di timeout calcolato (in secondi) e passa la variabile all'eseguibile del test case. Puoi leggere il valore della variabile per impostare un timer appropriato.

Interrompere

Si verifica quando il test runner interrompe l'IDT. Ad esempio, premendoCtrl+C.

Poiché i terminali propagano i segnali a tutti i processi secondari, è sufficiente configurare un gestore di segnali nei casi di test per rilevare i segnali di interruzione.

In alternativa, puoi interrogare periodicamente l'API per verificare il valore del`CancellationRequested` booleano nella risposta dell'`PollForNotifications`API.

Quando IDT riceve un segnale di interruzione, imposta il valore del`CancellationRequested` booleano su`true`.

Stop al primo fallimento

Si verifica quando un test case in esecuzione in parallel al test case corrente fallisce e il test runner utilizza l'`stop-on-first-failure`argomento per specificare che IDT deve interrompersi in caso di errore.

Per rilevare questo evento, puoi periodicamente interrogare l'API per verificare il valore del`CancellationRequested` booleano nella risposta dell'`PollForNotifications`API. Quando IDT rileva un errore ed è configurato per interrompersi al primo errore, imposta il valore del`CancellationRequested` booleano su`true`.

Quando si verifica uno di questi eventi, IDT attende 5 minuti affinché tutti i test case attualmente in esecuzione finiscano l'esecuzione. Se tutti i test case in corso non si concludono entro 5 minuti, IDT forza l'arresto di ciascuno dei relativi processi. Se IDT non ha ricevuto i risultati dei test prima della fine dei processi, contrassegnerà i casi di test come scaduti. Come buona pratica, dovresti assicurarti che i tuoi casi di test eseguano le seguenti azioni quando incontrano uno degli eventi:

1. Smetti di eseguire la normale logica di test.
2. Pulisci tutte le risorse temporanee, ad esempio gli artefatti di test sul dispositivo sottoposto a test.
3. Segnala un risultato del test a IDT, ad esempio un errore o un errore del test.
4. Uscita.

Usa il contesto IDT

Quando IDT esegue una suite di test, la suite di test può accedere a un set di dati che è possibile utilizzare per determinare l'esecuzione di ciascun test. Questi dati sono chiamati contesto IDT. Ad esempio, la configurazione dei dati utente fornita dai test runner in un`userdata.json`il file è reso disponibile per le suite di test nel contesto IDT.

Il contesto IDT può essere considerato un documento JSON di sola lettura. Le suite di test possono recuperare dati e scrivere dati nel contesto utilizzando tipi di dati JSON standard come oggetti, array, numeri e così via.

Schema del contesto

Il contesto IDT utilizza il seguente formato:

```
{
    "config": {
        <config-json-content>
        "timeoutMultiplier": timeout-multiplier
    },
    "device": {
        <device-json-device-element>
    },
    "devicePool": {
        <device-json-pool-element>
    },
    "resource": {
        "devices": [
            {
                <resource-json-device-element>
                "name": "<resource-name>"
            }
        ]
    },
    "testData": {
        "awsCredentials": {
            "awsAccessKeyId": "<access-key-id>",
            "awsSecretAccessKey": "<secret-access-key>",
            "awsSessionToken": "<session-token>"
        },
        "logFilePath": "/path/to/log/file"
    },
    "userData": {
        <userdata-json-content>
    }
}
```

`config`

Informazioni dal`config.json`documento. La`config`field contiene anche il seguente campo aggiuntivo:

`config.timeoutMultiplier`

Il moltiplicatore per il valore di qualsiasi timeout utilizzato dalla suite di test. Questo valore è specificato dal test runner della CLI IDT. Il valore di default è 1.

`device`

Informazioni sul dispositivo selezionato per l'esecuzione di test. Queste informazioni sono equivalenti al`devices`elemento array nel`device.json`documentoper il dispositivo selezionato.

`devicePool`

Informazioni sul pool di dispositivi selezionato per l'esecuzione del test. Queste informazioni sono equivalenti all'elemento array del pool di dispositivi di livello superiore definito nella`device.json`file per il pool di dispositivi selezionato.

`resource`

Informazioni sui dispositivi di risorse dal`resource.json`file.

`resource.devices`

Queste informazioni sono equivalenti al`devices`array definito nel`resource.json`file. Ciascuno`devices`include il seguente campo aggiuntivo:

`resource.device.name`

Il nome del dispositivo delle risorse. Questo valore è impostato sul`requiredResource.name`nel`test.json`file.

`testData.awsCredentials`

LaAWScredenziali utilizzate dal test per connettersi alAWSnuvola. Queste informazioni sono ottenute dal`config.json`file.

`testData.logFilePath`

Il percorso del file di log in cui il test case scrive i messaggi di log. Se non esiste, la suite di test crea questo file.

`userData`

Informazioni fornite dal corridore di prova nella`userdata.json`documento.

Accesso ai dati nel contesto

È possibile eseguire query sul contesto utilizzando la notazione JSONPath dai file JSON e dal file eseguibile di testo con il`GetContextValue`e`GetContextString`API. La sintassi per le stringhe JsonPath per accedere al contesto IDT varia come segue:

- Nello stato`suite.jsone`test.json, utilizzi{{*query*}}. Cioè, non utilizzare l'elemento radice`$.`per iniziare la tua espressione.
- Nello stato`test_orchestrator.yaml`, utilizzi{{*query*}}.

  Se si utilizza la macchina a stato deprecato, quindi in`state_machine.json`, utilizzi{{`$.`*query*}}.
- Nei comandi API, utilizzi*query*o{{`$.`*query*}}, a seconda del comando. Per ulteriori informazioni, consulta la documentazione in linea negli SDK.

Nella tabella seguente vengono descritti gli operatori di un'espressione JSONPath tipica:

Operator	Description
`$`	The root element. Because the top-level context value for IDT is an object, you will typically use `$.` to start your queries.
`.ChildName`	Accesses the child element with name `Nome figlio` from an object. If applied to an array, yields a new array with this operator applied to each element. The element name is case sensitive. For example, the query to access the `awsRegion` value in the `config` object is `Regione $.config.aws` .
`[inizio:fine]`	Filters elements from an array, retrieving items beginning from the `avvio` index and going up to the `fine` index, both inclusive.
`[index1, index2,..., indexN]`	Filters elements from an array, retrieving items from only the specified indices.
`[? (expr)]`	Filters elements from an array using the `expr` expression. This expression must evaluate to a boolean value.

Per creare espressioni di filtro, utilizzare la seguente sintassi:

```
<jsonpath> | <value> operator <jsonpath> | <value>
```

In questa sintassi:

- `jsonpath`è un JSONPath che utilizza la sintassi JSON standard.
- `value`è un valore personalizzato che utilizza la sintassi JSON standard.
- `operator`è uno dei seguenti operatori:
  - <(Minore di)
  - <=(Minore di o uguale a)
  - ==(Equal to)

    Se il valore o JsonPath nell'espressione è un valore di array, booleano o oggetto, questo è l'unico operatore binario supportato che è possibile utilizzare.
  - >=(Maggiore di o uguale a)
  - >(Maggiore di)
  - =~(Corrispondenza dell'espressione regolare). Per utilizzare questo operatore in un'espressione di filtro, il valore JsonPath o sul lato sinistro dell'espressione deve essere valutato in una stringa e il lato destro deve essere un valore di pattern che segue ilSintassi RE2.

È possibile utilizzare le query JsonPath nel modulo {{*domanda*}} come stringhe segnaposto all'interno del`argse`del`environmentVariables`campi in`test.json`file e all'interno del`environmentVariables`campi in`suite.json`file. IDT esegue una ricerca di contesto e popola i campi con il valore valutato della query. Ad esempio, nel`suite.json`file, è possibile utilizzare stringhe segnaposto per specificare i valori delle variabili di ambiente che cambiano con ogni test case e IDT popolerà le variabili di ambiente con il valore corretto per ogni test case. Tuttavia, quando si utilizzano stringhe segnaposto in`test.json`e`suite.json`, per le tue query si applicano le seguenti considerazioni:

- È necessario ogni occorrenza del`devicePool`chiave nella tua query in minuscolo. Cioè, usa`devicepool`invece.
- Per gli array, è possibile utilizzare solo array di stringhe. Inoltre, gli array utilizzano uno standard non standard`item1, item2,...,item`N. Se l'array contiene un solo elemento, viene serializzato come`item`, rendendolo indistinguibile da un campo stringa.
- Non è possibile utilizzare i segnaposto per recuperare oggetti dal contesto.

A causa di queste considerazioni, ti consigliamo di utilizzare l'API per accedere al contesto nella logica di test invece di stringhe segnaposto in`test.jsonesuite.json`file. Tuttavia, in alcuni casi potrebbe essere più conveniente utilizzare i segnaposto JsonPath per recuperare stringhe singole da impostare come variabili di ambiente.

Configurare le impostazioni per i test runner

Per eseguire suite di test personalizzate, i test runner devono configurare le proprie impostazioni in base alla suite di test che desiderano eseguire. Le impostazioni vengono specificate in base ai modelli di file di configurazione presenti nella *<device-tester-extract-location>*`/configs/` cartella. Se necessario, i test runner devono anche impostare AWS le credenziali che IDT utilizzerà per connettersi al cloud. AWS

In qualità di scrittore di test, dovrai configurare questi file per eseguire il debug della tua suite di test. È necessario fornire istruzioni ai test runner in modo che possano configurare le seguenti impostazioni in base alle esigenze per eseguire le suite di test.

Configura dispositivo.json

Il `device.json` file contiene informazioni sui dispositivi su cui vengono eseguiti i test (ad esempio, indirizzo IP, informazioni di accesso, sistema operativo e architettura della CPU).

I test runner possono fornire queste informazioni utilizzando il seguente `device.json` file modello che si trova nella *<device-tester-extract-location>*`/configs/` cartella.

```
[
    {
        "id": "<pool-id>",
        "sku": "<pool-sku>",
        "features": [
            {
                "name": "<feature-name>",
                "value": "<feature-value>",
                "configs": [
                    {
                        "name": "<config-name>",
                        "value": "<config-value>"
                    }
                ],
            }
        ],
        "devices": [
```

```
            {
                "id": "<device-id>",
                "connectivity": {
                    "protocol": "ssh | uart | docker",
                    // ssh
                    "ip": "<ip-address>",
                    "port": <port-number>,
                    "auth": {
                        "method": "pki | password",
                        "credentials": {
                            "user": "<user-name>",
                            // pki
                            "privKeyPath": "/path/to/private/key",

                            // password
                            "password": "<password>",
                        }
                    },

                    // uart
                    "serialPort": "<serial-port>",

                    // docker
                    "containerId": "<container-id>",
                    "containerUser": "<container-user-name>",
                }
            }
        ]
    }
]
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`id`

Un ID alfanumerico definito dall'utente che identifica in modo univoco una raccolta di dispositivi denominata un pool di dispositivi. I dispositivi che appartengono a un pool devono avere lo stesso hardware. Durante l'esecuzione di una suite di test, i dispositivi del pool vengono utilizzati per parallelizzare il carico di lavoro. Più dispositivi vengono utilizzati per eseguire diversi test.

`sku`

Un valore alfanumerico che identifica in modo univoco il dispositivo sottoposto a test. Lo SKU viene utilizzato per tracciare i dispositivi qualificati.

> **Note**
>
> Se vuoi elencare la scheda in AWS Partner Device Catalog, il codice SKU specificato qui deve corrispondere al codice SKU utilizzato nel processo di elencazione.

`features`

Facoltativo. Un array contenente le caratteristiche supportate del dispositivo. Le funzionalità del dispositivo sono valori definiti dall'utente che configuri nella tua suite di test. È necessario fornire ai test runner informazioni sui nomi e sui valori delle funzionalità da includere nel `device.json` file. Ad esempio, se desiderate testare un dispositivo che funge da server MQTT per altri dispositivi, potete configurare la logica di test per convalidare livelli supportati specifici per una funzionalità denominata. `MQTT_QOS` I test runner forniscono questo nome di funzionalità e impostano il valore della funzionalità sui livelli QOS supportati dal proprio dispositivo. È possibile recuperare le informazioni fornite dal contesto IDT con la `devicePool.features` query o dal contesto di test orchestrator con la query. `pool.features`

`features.name`

Il nome della funzionalità.

`features.value`

I valori delle funzionalità supportate.

`features.configs`

Impostazioni di configurazione, se necessarie, per la funzionalità.

`features.config.name`

Il nome dell'impostazione di configurazione.

`features.config.value`

I valori di impostazione supportati.

`devices`

Una serie di dispositivi nel pool da testare. È richiesto almeno un dispositivo.

`devices.id`

Un identificativo univoco definito dall'utente del dispositivo sottoposto a test.

`connectivity.protocol`

Il protocollo di comunicazione utilizzato per comunicare con questo dispositivo. Ogni dispositivo in un pool deve utilizzare lo stesso protocollo.

Attualmente, gli unici valori supportati sono `ssh` e `uart` per i dispositivi fisici e `docker` per i contenitori Docker.

`connectivity.ip`

L'indirizzo IP del dispositivo sottoposto a test.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.port`

Facoltativo. Il numero di porta da utilizzare per le connessioni SSH.

Il valore predefinito è 22.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.auth`

Informazioni di autenticazione per la connessione.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.auth.method`

Il metodo di autorizzazione utilizzato per accedere a un dispositivo con un determinato protocollo di connettività.

I valori supportati sono:

- `pki`
- `password`

`connectivity.auth.credentials`

Le credenziali utilizzate per l'autenticazione.

`connectivity.auth.credentials.password`

La password utilizzata per l'accesso al dispositivo da testare.

Questo valore si applica solo se `connectivity.auth.method` è impostato su `password`.

`connectivity.auth.credentials.privKeyPath`

Il percorso completo alla chiave privata utilizzata per accedere al dispositivo sottoposto a test.

Questo valore si applica solo se `connectivity.auth.method` è impostato su `pki`.

`connectivity.auth.credentials.user`

Il nome utente per l'accesso al dispositivo sottoposto a test.

`connectivity.serialPort`

Facoltativo. La porta seriale a cui è collegato il dispositivo.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `uart`.

`connectivity.containerId`

L'ID contenitore o il nome del contenitore Docker in fase di test.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.containerUser`

Facoltativo. Il nome da utente a utente all'interno del contenitore. Il valore predefinito è l'utente fornito nel Dockerfile.

Il valore predefinito è 22.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

> **Note**
>
> Per verificare se i test runner configurano la connessione errata del dispositivo per un test, puoi recuperarlo `pool.Devices[0].Connectivity.Protocol` dal contesto del test orchestrator e confrontarlo con il valore previsto in uno stato. `Choice` Se viene utilizzato un protocollo errato, stampa un messaggio utilizzando `LogMessage` lo stato e passa allo stato. `Fail`
> In alternativa, è possibile utilizzare il codice di gestione degli errori per segnalare un errore di test per tipi di dispositivi errati.

(Facoltativo) Configura userdata.json

Il `userdata.json` file contiene tutte le informazioni aggiuntive richieste da una suite di test ma non specificate nel file. `device.json` Il formato di questo file dipende dal `userdata_scheme.json`file definito nella suite di test. Se sei uno scrittore di test, assicurati di fornire queste informazioni agli utenti che eseguiranno le suite di test che scrivi.

(Facoltativo) Configura resource.json

Il `resource.json` file contiene informazioni su tutti i dispositivi che verranno utilizzati come dispositivi di risorse. I dispositivi di risorse sono dispositivi necessari per testare determinate funzionalità di un dispositivo sottoposto a test. Ad esempio, per testare la funzionalità Bluetooth di un dispositivo, è possibile utilizzare un dispositivo di risorse per verificare che il dispositivo sia in grado di connettersi correttamente ad esso. I dispositivi di risorse sono opzionali e puoi richiedere tutti i dispositivi di risorse di cui hai bisogno. In qualità di autore del test, utilizzi il file test.json per definire le funzionalità dei dispositivi di risorse necessarie per un test. I test runner utilizzano quindi il `resource.json` file per fornire un pool di dispositivi di risorse dotati delle funzionalità richieste. Assicurati di fornire queste informazioni agli utenti che eseguiranno le suite di test che scrivi.

I test runner possono fornire queste informazioni utilizzando il seguente `resource.json` file modello che si trova nella *<device-tester-extract-location>*`/configs/` cartella.

```
[
    {
        "id": "<pool-id>",
        "features": [
            {
                "name": "<feature-name>",
                "version": "<feature-version>",
                "jobSlots": <job-slots>
            }
        ],
        "devices": [
            {
                "id": "<device-id>",
                "connectivity": {
                    "protocol": "ssh | uart | docker",
                    // ssh
                    "ip": "<ip-address>",
                    "port": <port-number>,
                    "auth": {
```

```
                        "method": "pki | password",
                        "credentials": {
                            "user": "<user-name>",
                            // pki
                            "privKeyPath": "/path/to/private/key",

                            // password
                            "password": "<password>",
                        }
                    },

                    // uart
                    "serialPort": "<serial-port>",

                    // docker
                    "containerId": "<container-id>",
                    "containerUser": "<container-user-name>",
                }
            }
        ]
    }
]
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

`id`

Un ID alfanumerico definito dall'utente che identifica in modo univoco una raccolta di dispositivi denominata un pool di dispositivi. I dispositivi che appartengono a un pool devono avere lo stesso hardware. Durante l'esecuzione di una suite di test, i dispositivi del pool vengono utilizzati per parallelizzare il carico di lavoro. Più dispositivi vengono utilizzati per eseguire diversi test.

`features`

Facoltativo. Un array contenente le caratteristiche supportate del dispositivo. Le informazioni richieste in questo campo sono definite nei file test.json nella suite di test e determinano quali test eseguire e come eseguirli. Se la suite di test non richiede alcuna funzionalità, questo campo non è obbligatorio.

`features.name`

Il nome della funzionalità.

`features.version`

La versione della funzionalità.

`features.jobSlots`

Impostazione per indicare quanti test possono utilizzare il dispositivo contemporaneamente. Il valore predefinito è 1.

`devices`

Una serie di dispositivi nel pool da testare. È richiesto almeno un dispositivo.

`devices.id`

Un identificativo univoco definito dall'utente del dispositivo sottoposto a test.

`connectivity.protocol`

Il protocollo di comunicazione utilizzato per comunicare con questo dispositivo. Ogni dispositivo in un pool deve utilizzare lo stesso protocollo.

Attualmente, gli unici valori supportati sono `ssh` e `uart` per i dispositivi fisici e `docker` per i contenitori Docker.

`connectivity.ip`

L'indirizzo IP del dispositivo sottoposto a test.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.port`

Facoltativo. Il numero di porta da utilizzare per le connessioni SSH.

Il valore predefinito è 22.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.auth`

Informazioni di autenticazione per la connessione.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.auth.method`

Il metodo di autorizzazione utilizzato per accedere a un dispositivo con un determinato protocollo di connettività.

I valori supportati sono:

- `pki`
- `password`

`connectivity.auth.credentials`

Le credenziali utilizzate per l'autenticazione.

`connectivity.auth.credentials.password`

La password utilizzata per l'accesso al dispositivo da testare.

Questo valore si applica solo se `connectivity.auth.method` è impostato su `password`.

`connectivity.auth.credentials.privKeyPath`

Il percorso completo alla chiave privata utilizzata per accedere al dispositivo sottoposto a test.

Questo valore si applica solo se `connectivity.auth.method` è impostato su `pki`.

`connectivity.auth.credentials.user`

Il nome utente per l'accesso al dispositivo sottoposto a test.

`connectivity.serialPort`

Facoltativo. La porta seriale a cui è collegato il dispositivo.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `uart`.

`connectivity.containerId`

L'ID contenitore o il nome del contenitore Docker in fase di test.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

`connectivity.containerUser`

Facoltativo. Il nome da utente a utente all'interno del contenitore. Il valore predefinito è l'utente fornito nel Dockerfile.

Il valore predefinito è 22.

Questa proprietà si applica solo se `connectivity.protocol` è impostata su `ssh`.

(Facoltativo) Configura config.json

Il `config.json` file contiene informazioni di configurazione per IDT. In genere, i test runner non avranno bisogno di modificare questo file se non per fornire le proprie credenziali AWS utente per IDT e, facoltativamente, una regione. AWS Se vengono fornite AWS le credenziali con le autorizzazioni richieste, AWS IoT Device Tester raccoglie e invia le metriche di utilizzo a. AWS Si tratta di una funzionalità opzionale e viene utilizzata per migliorare la funzionalità IDT. Per ulteriori informazioni, consulta Parametri di utilizzo IDT.

I test runner possono configurare le proprie AWS credenziali in uno dei seguenti modi:

- File di credenziali

  IDT usa lo stesso file delle credenziali di AWS CLI. Per ulteriori informazioni, consulta l'argomento relativo ai file di configurazione e delle credenziali.

  La posizione del file delle credenziali varia in base al sistema operativo in uso:

  - macOS, Linux: `~/.aws/credentials`
  - Windows: `C:\Users\`*`UserName`*`\.aws\credentials`
- Variabili di ambiente

  Le variabili di ambiente sono variabili gestite dal sistema operativo e utilizzate dai comandi di sistema. Le variabili definite durante una sessione SSH non sono disponibili dopo la chiusura della sessione. IDT può utilizzare le variabili di `AWS_SECRET_ACCESS_KEY` ambiente `AWS_ACCESS_KEY_ID` e per memorizzare le credenziali AWS

  Per impostare queste variabili su Linux, macOS o Unix, utilizza export:

  ```
  export AWS_ACCESS_KEY_ID=<your_access_key_id>
  export AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
  ```

  Per impostare queste variabili su Windows, utilizza set:

  ```
  set AWS_ACCESS_KEY_ID=<your_access_key_id>
  set AWS_SECRET_ACCESS_KEY=<your_secret_access_key>
  ```

Per configurare AWS le credenziali per IDT, i test runner modificano la `auth` sezione del `config.json` file che si trova nella cartella. *`<device-tester-extract-location>`*`/configs/`

```
{
    "log": {
        "location": "logs"
    },
    "configFiles": {
        "root": "configs",
        "device": "configs/device.json"
    },
    "testPath": "tests",
    "reportPath": "results",
    "awsRegion": "<region>",
    "auth": {
        "method": "file | environment",
        "credentials": {
            "profile": "<profile-name>"
        }
    }
}
]
```

Tutti i campi che includono valori sono obbligatori, come descritto di seguito:

> **Note**
>
> *Tutti i percorsi di questo file sono definiti in relazione a < >. device-tester-extract-location*

`log.location`

Il percorso della cartella logs nel file *< device-tester-extract-location >*.

`configFiles.root`

Il percorso della cartella che contiene i file di configurazione.

`configFiles.device`

Il percorso del `device.json` file.

`testPath`

Il percorso della cartella che contiene le suite di test.

`reportPath`

Il percorso della cartella che conterrà i risultati dei test dopo che IDT avrà eseguito una suite di test.

`awsRegion`

Facoltativo. La AWS regione che verranno utilizzate dalle suite di test. Se non è impostata, le suite di test utilizzeranno la regione predefinita specificata in ciascuna suite di test.

`auth.method`

Il metodo utilizzato da IDT per recuperare AWS le credenziali. I valori supportati sono `file` il recupero delle credenziali da un file di credenziali e il recupero delle credenziali utilizzando le variabili `environment` di ambiente.

`auth.credentials.profile`

Il profilo delle credenziali da utilizzare dal file delle credenziali. Questa proprietà si applica solo se `auth.method` è impostata su `file`.

Esegui il debug ed esegui suite di test personalizzate

Dopo aver impostato la configurazione richiesta, IDT può eseguire la suite di test. Il tempo di esecuzione della suite di test completa dipende dall'hardware e dalla composizione della suite di test. Per riferimento, sono necessari circa 30 minuti per completare l'intera suite di test di AWS IoT Greengrass qualificazione su un Raspberry Pi 3B.

Durante la scrittura della suite di test, è possibile utilizzare IDT per eseguire la suite di test in modalità debug per controllare il codice prima di eseguirlo o fornirlo ai test runner.

Esegui IDT in modalità di debug

Poiché le suite di test dipendono da IDT per interagire con i dispositivi, fornire il contesto e ricevere risultati, non è possibile semplicemente eseguire il debug delle suite di test in un IDE senza alcuna interazione IDT. A tale scopo, la CLI IDT fornisce `debug-test-suite` il comando che consente di eseguire IDT in modalità di debug. Eseguite il comando seguente per visualizzare le opzioni disponibili per: `debug-test-suite`

```
devicetester_[linux | mac | win_x86-64] debug-test-suite -h
```

Quando eseguite IDT in modalità debug, IDT non avvia effettivamente la suite di test né esegue il test orchestrator; interagisce invece con l'IDE per rispondere alle richieste fatte dalla suite di test in esecuzione nell'IDE e stampa i log sulla console. IDT non scade e attende di uscire fino a quando non viene interrotto manualmente. In modalità debug, IDT inoltre non esegue il test orchestrator e non genererà alcun file di report. Per eseguire il debug della suite di test, è necessario utilizzare l'IDE per fornire alcune informazioni che IDT di solito ottiene dai file JSON di configurazione. Assicurati di fornire le seguenti informazioni:

- Variabili di ambiente e argomenti per ogni test. IDT non leggerà queste informazioni da `test.json` o`suite.json`.
- Argomenti per selezionare i dispositivi di risorse. IDT non leggerà queste informazioni da`test.json`.

Per eseguire il debug delle tue suite di test, completa i seguenti passaggi:

1. Crea i file di configurazione delle impostazioni necessari per eseguire la suite di test. Ad esempio, se la tua suite di test richiede `device.json` `resource.json`e `user data.json`, assicurati di configurarli tutti secondo necessità.
2. Eseguite il comando seguente per mettere IDT in modalità debug e selezionare tutti i dispositivi necessari per eseguire il test.

   ```
   devicetester_[linux | mac | win_x86-64] debug-test-suite [options]
   ```

   Dopo aver eseguito questo comando, IDT attende le richieste dalla suite di test e quindi risponde ad esse. IDT genera anche le variabili di ambiente necessarie per il processo di elaborazione dei casi per IDT Client SDK.
3. Nel tuo IDE, usa la `debug` configurazione `run` o per effettuare le seguenti operazioni:

   a. Imposta i valori delle variabili di ambiente generate da IDT.

   b. Imposta il valore di tutte le variabili o gli argomenti di ambiente che hai specificato nel file `test.json` and`suite.json`.

   c. Imposta i punti di interruzione in base alle esigenze.
4. Esegui la suite di test nel tuo IDE.

   Puoi eseguire il debug e rieseguire la suite di test tutte le volte che è necessario. IDT non scade in modalità di debug.

5. Dopo aver completato il debug, interrompi IDT per uscire dalla modalità di debug.

Comandi IDT CLI per eseguire test

La sezione seguente descrive i comandi IDT CLI:

IDT v4.0.0

`help`

Elenca le informazioni sul comando specificato.

`list-groups`

Elenca i gruppi in una determinata suite di test.

`list-suites`

Elenca le suite di test disponibili.

`list-supported-products`

Elenca i prodotti supportati per la versione di IDT in uso, in questo caso AWS IoT Greengrass le versioni, e le versioni della suite di test di AWS IoT Greengrass qualificazione disponibili per la versione IDT corrente.

`list-test-cases`

Elenca i casi di test in un determinato gruppo di test. È supportata la seguente opzione:

- `group-id`. Il gruppo di test da cercare. Questa opzione è obbligatoria e deve specificare un singolo gruppo.

`run-suite`

Esegue una suite di test in un determinato pool di dispositivi. Di seguito sono riportate alcune opzioni di uso comune:

- `suite-id`. La versione della suite di test da eseguire. Se non specificato, IDT utilizza la versione più recente nella cartella `tests`.
- `group-id`. I gruppi di test da eseguire, sotto forma di elenco separato da virgole. Se non specificato, IDT esegue tutti i gruppi di test nella suite di test.
- `test-id`. I casi di test da eseguire, come elenco separato da virgole. Quando specificato, `group-id` deve specificare un singolo gruppo.

- `pool-id`. Il pool di dispositivi da testare. I test runner devono specificare un pool se hanno più pool di dispositivi definiti nel `device.json` file.
- `timeout-multiplier`. Configura IDT per modificare il timeout di esecuzione del test specificato nel `test.json` file per un test con un moltiplicatore definito dall'utente.
- `stop-on-first-failure`. Configura IDT per interrompere l'esecuzione al primo errore. Questa opzione deve essere utilizzata con `group-id` per eseguire il debug dei gruppi di test specificati.
- `userdata`. Imposta il file che contiene le informazioni sui dati utente necessarie per eseguire la suite di test. Questo è richiesto solo se `userdataRequired` è impostato su true nel `suite.json` file della suite di test.

Per ulteriori informazioni sulle opzioni `run-suite`, utilizzare l'opzione `help`:

```
devicetester_[linux | mac | win_x86-64] run-suite -h
```

`debug-test-suite`

Esegui la suite di test in modalità debug. Per ulteriori informazioni, consulta Esegui IDT in modalità di debug.

Rivedi i risultati e i registri dei test

Questa sezione descrive il formato in cui IDT genera registri della console e report di test.

Formato dei messaggi della console

AWS IoT Device Testerutilizza un formato standard per la stampa di messaggi sulla console quando avvia una suite di test. Di seguito viene riportato un estratto di esempio di messaggio della console generato da IDT.

```
time="2000-01-02T03:04:05-07:00" level=info msg=Using suite: MyTestSuite_1.0.0
executionId=9a52f362-1227-11eb-86c9-8c8590419f30
```

La maggior parte dei messaggi della console sono costituiti dai seguenti campi:

`time`

Un timestamp ISO 8601 completo per l'evento registrato.

`level`

Il livello del messaggio per l'evento registrato. In genere, il livello di messaggio registrato è uno dei`info`,`warn`, oppure`error`. I componenti IDT emette`fatal`o`panic`messaggio se incontra un evento previsto che lo fa uscire in anticipo.

`msg`

Il messaggio registrato.

`executionId`

Stringa ID univoca per il processo IDT corrente. Questo ID viene utilizzato per distinguere tra singole esecuzioni IDT.

I messaggi della console generati da una suite di test forniscono informazioni aggiuntive sul dispositivo in fase di test e sulla suite di test, sul gruppo di test e sui casi di test eseguiti da IDT. Il seguente estratto mostra un esempio di messaggio della console generato da una suite di test.

```
time="2000-01-02T03:04:05-07:00" level=info msg=Hello world! suiteId=MyTestSuite
groupId=myTestGroup testCaseId=myTestCase deviceId=my-device
executionId=9a52f362-1227-11eb-86c9-8c8590419f30
```

La parte specifica della suite di test del messaggio della console contiene i seguenti campi:

`suiteId`

Il nome della suite di test attualmente in esecuzione.

`groupId`

L'ID del gruppo di test attualmente in esecuzione.

`testCaseId`

La corrente del caso di test in esecuzione.

`deviceId`

ID del dispositivo sottoposto a test utilizzato dal test case corrente.

Per stampare un riepilogo di test sulla console quando un IDT termina l'esecuzione di un test, è necessario includere un`Reportstate`nel tuo orchestratore di test. Il riepilogo del test contiene

informazioni sulla suite di test, i risultati del test per ciascun gruppo eseguito e le posizioni dei log e dei file di report generati. Il seguente esempio mostra un messaggio di riepilogo dei test.

```
========== Test Summary ==========
Execution Time:     5m00s
Tests Completed:    4
Tests Passed:       3
Tests Failed:       1
Tests Skipped:      0
---------------------------------
Test Groups:
    GroupA:         PASSED
    GroupB:         FAILED
---------------------------------
Failed Tests:
    Group Name: GroupB
        Test Name: TestB1
            Reason: Something bad happened
---------------------------------
Path to IoT Device Tester Report: /path/to/awsiotdevicetester_report.xml
Path to Test Execution Logs: /path/to/logs
Path to Aggregated JUnit Report: /path/to/MyTestSuite_Report.xml
```

AWS IoT Device Testerschema dei report

`awsiotdevicetester_report.xml`Un report firmato contenente le seguenti informazioni:

- La versione di IDT.
- La versione della suite di test.
- La firma e la chiave del report utilizzati per firmare il report.
- Il codice SKU del dispositivo e il nome del pool di dispositivi specificato nella`device.json`file.
- La versione del prodotto e le caratteristiche del dispositivo testate.
- Il riepilogo aggregato dei risultati dei test. Queste informazioni sono le stesse di quelle contenute nel*`suite-name`*`_report.xml`file.

```
<apnreport>
    <awsiotdevicetesterversion>idt-version</awsiotdevicetesterversion>
    <testsuiteversion>test-suite-version</testsuiteversion>
    <signature>signature</signature>
```

```
    <keyname>keyname</keyname>
    <session>
        <testsession>execution-id</testsession>
        <starttime>start-time</starttime>
        <endtime>end-time</endtime>
    </session>
    <awsproduct>
        <name>product-name</name>
        <version>product-version</version>
        <features>
            <feature name="<feature-name>" value="supported | not-supported | <feature-
value>" type="optional | required"/>
        </features>
    </awsproduct>
    <device>
        <sku>device-sku</sku>
        <name>device-name</name>
        <features>
            <feature name="<feature-name>" value="<feature-value>"/>
        </features>
        <executionMethod>ssh | uart | docker</executionMethod>
    </device>
    <devenvironment>
        <os name="<os-name>"/>
    </devenvironment>
    <report>
        <suite-name-report-contents>
    </report>
</apnreport>
```

Il file `awsiotdevicetester_report.xml` contiene un tag `<awsproduct>` con le informazioni relative al prodotto sottoposto a test e le caratteristiche del prodotto che sono state convalidate dopo l'esecuzione di una suite di test.

Attributi utilizzati nel tag **`<awsproduct>`**

`name`

Il nome del prodotto sottoposto a test.

`version`

La versione del prodotto sottoposto a test.

`features`

Le caratteristiche convalidate. Caratteristiche contrassegnate come`required`sono necessari per la suite di test per convalidare il dispositivo. Il seguente frammento di codice mostra come questa informazione viene visualizzata nel file `awsiotdevicetester_report.xml`.

```
<feature name="ssh" value="supported" type="required"></feature>
```

Caratteristiche contrassegnate come`optional`non sono necessari per la convalida. I seguenti snippet mostrano caratteristiche facoltative.

```
<feature name="hsi" value="supported" type="optional"></feature>

<feature name="mqtt" value="not-supported" type="optional"></feature>
```

Schema dei report della suite di

Il report *`suite-name`*`_Result.xml` è in formato XML JUnit. Puoi eseguire l'integrazione in piattaforme di integrazione e distribuzione continue come Jenkins, Bambù e così via. Il report contiene un riepilogo aggregato dei risultati dei test.

```
<testsuites name="<suite-name> results" time="<run-duration>" tests="<number-of-test>"
 failures="<number-of-tests>" skipped="<number-of-tests>" errors="<number-of-tests>"
 disabled="0">
    <testsuite name="<test-group-id>" package="" tests="<number-of-tests>"
 failures="<number-of-tests>" skipped="<number-of-tests>" errors="<number-of-tests>"
 disabled="0">
        <!--success-->
        <testcase classname="<classname>" name="<name>" time="<run-duration>"/>
        <!--failure-->
        <testcase classname="<classname>" name="<name>" time="<run-duration>">
            <failure type="<failure-type>">
                reason
            </failure>
        </testcase>
        <!--skipped-->
        <testcase classname="<classname>" name="<name>" time="<run-duration>">
            <skipped>
                reason
            </skipped>
```

```
        </testcase>
        <!--error-->
        <testcase classname="<classname>" name="<name>" time="<run-duration>">
            <error>
                reason
            </error>
        </testcase>
    </testsuite>
</testsuites>
```

La sezione report in entrambi i`awsiotdevicetester_report.xml`o*suite-name*`_report.xml`Elenca i test eseguiti e i risultati.

Il primo tag XML `<testsuites>` contiene il riepilogo dell'esecuzione dei test. Ad esempio:

```
<testsuites name="MyTestSuite results" time="2299" tests="28" failures="0" errors="0"
 disabled="0">
```

Attributi utilizzati nel tag **`<testsuites>`**

`name`

Il nome della suite di test.

`time`

Il tempo, espresso in secondi, impiegato per eseguire la suite di test.

`tests`

Il numero di test eseguiti.

`failures`

Il numero di test eseguiti ma non superati.

`errors`

Il numero di test che IDT non è stato in grado di eseguire.

`disabled`

Questo attributo non è utilizzato e si può ignorare.

In caso di esiti negativi o errori nei test, puoi identificare il test non riuscito esaminando i tag XML `<testsuites>`. I tag XML `<testsuite>` all'interno del tag `<testsuites>` mostrano il riepilogo dei risultati dei test per un gruppo di test. Ad esempio:

```
<testsuite name="combination" package="" tests="1" failures="0" time="161" disabled="0"
 errors="0" skipped="0">
```

Il formato è simile al tag `<testsuites>`, ma con un attributo `skipped` che non viene utilizzato e che è possibile ignorare. All'interno di ogni tag XML `<testsuite>` ci sono tag `<testcase>` per ciascuno dei test eseguiti per un gruppo di test. Ad esempio:

```
<testcase classname="Security Test" name="IP Change Tests" attempts="1"></testcase>>
```

Attributi utilizzati nel tag **`<testcase>`**

`name`

Il nome del test.

`attempts`

Il numero di volte che IDT ha eseguito il test.

Quando un test non riesce o si verifica un errore, i tag `<failure>` o `<error>` vengono aggiunti al tag `<testcase>` con informazioni per la risoluzione dei problemi. Ad esempio:

```
<testcase classname="mcu.Full_MQTT" name="MQTT_TestCase" attempts="1">
 <failure type="Failure">Reason for the test failure</failure>
 <error>Reason for the test execution error</error>
</testcase>
```

Parametri di utilizzo IDT

Se fornisciAWS le credenziali con le autorizzazioni richieste,AWS IoT Device Tester raccoglie e invia le metriche di utilizzo aAWS. Questa è una funzionalità opzionale e viene utilizzata per migliorare la funzionalità IDT. IDT raccoglie informazioni come le seguenti:

- L'Account AWSID utilizzato per eseguire IDT
- IAWS CLI comandi IDT usati per eseguire i test

- Le suite di test che vengono eseguite
- Le suite di test nella cartella *<device-tester-extract-location >*
- Il numero di dispositivi configurati nel pool di dispositivi
- Nomi e tempi di esecuzione dei test case
- Informazioni sui risultati del test, ad esempio se i test sono stati superati, non riusciti, hanno riscontrato errori o sono stati ignorati
- Caratteristiche del prodotto testate
- Comportamento di uscita IDT, ad esempio uscite inaspettate o anticipate

Tutte le informazioni inviate da IDT vengono inoltre registrate in un`metrics.log` file nella*<device-tester-extract-location>*`/results/`*<execution-id>*`/` cartella. È possibile visualizzare il file di registro per visualizzare le informazioni raccolte durante un'esecuzione di test. Questo file viene generato solo se scegli di raccogliere le metriche di utilizzo.

Per disabilitare la raccolta delle metriche, non è necessario intraprendere ulteriori azioni. Semplicemente non memorizzateAWS le vostre credenziali e, se aveteAWS credenziali memorizzate, non configurate il`config.json` file per accedervi.

Configura le credenziali AWS

Se non disponi già di unoAccount AWS, devi crearne uno. Se ne hai già unoAccount AWS, devi semplicemente configurare le autorizzazioni richieste per il tuo account che consentono a IDT di inviare le metriche di utilizzo per tuoAWS conto.

Fase 1: creare un Account AWS

In questo passaggio, crea e configura unAccount AWS. Se disponi già di unAccount AWS, passa athe section called “Fase 2: configurazione delle autorizzazioni per IDT”.

Se non si dispone di un Account AWS, completare la procedura seguente per crearne uno.

Come registrarsi a un Account AWS

1. Aprire la pagina all'indirizzo https://portal.aws.amazon.com/billing/signup.
2. Seguire le istruzioni online.

   Nel corso della procedura di registrazione, si riceverà una telefonata, durante la quale sarà necessario inserire un codice di verifica sulla tastiera del telefono.

Durante la registrazione di un Account AWS, viene creato un Utente root dell'account AWS. L'utente root ha accesso a tutte le risorse e i Servizi AWS in tale account. Come best practice di sicurezza, assegnare l'accesso amministrativo a un utente amministrativo e utilizzare solo l'utente root per eseguire attività che richiedono l'accesso di un utente root.

Per creare un utente amministratore, scegli una delle seguenti opzioni.

Scelta di un modo per gestire il tuo amministratore	Per	By	Puoi anche
In IAM Identity Center (Consigliato)	Usa credenziali a breve termine per accedere a AWS. Ciò è in linea con le best practice per la sicurezza. Per informazioni sulle best practice, consulta Best practice per la sicurezza in IAM nella Guida per l'utente di IAM.	Segui le istruzioni riportate in Nozioni di base nella Guida per l'utente di AWS IAM Identity Center.	Configura l'accesso programmatico seguendo quanto riportato in Configurazione della AWS CLI per utilizzare AWS IAM Identity Center nella Guida per l'utente di AWS Command Line Interface.
In IAM (Non consigliato)	Usa credenziali a lungo termine per accedere a AWS.	Segui le istruzioni in Creazione del primo utente e gruppo di utenti IAM di amministrazione nella Guida per l'utente IAM.	Configura l'accesso programmatico seguendo quanto riportato in Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Fase 2: configurazione delle autorizzazioni per IDT

In questo passaggio, configura le autorizzazioni utilizzate da IDT per eseguire i test e raccogliere i dati di utilizzo dell'IDT. È possibile utilizzareAWS Management Console orAWS Command Line Interface (AWS CLI) per creare una policy IAM e un utente per IDT, quindi allegare policy all'utente.

- Per configurare le autorizzazioni per IDT (Console)
- Per configurare le autorizzazioni per IDT (AWS CLI)

Per configurare le autorizzazioni per IDT (Console)

Attenersi alla seguente procedura per utilizzare la console per configurare le autorizzazioni per IDT per AWS IoT Greengrass.

1. Accedi alla console IAM.
2. Creare un criterio gestito dal cliente che concede le autorizzazioni per creare ruoli con autorizzazioni specifiche.
   a. Nel riquadro di navigazione, selezionare Policies e scegliere Create Policy (Crea policy).
   b. Nella scheda JSON, sostituire il contenuto del segnaposto con la seguente policy.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot-device-tester:SendMetrics"
            ],
            "Resource": "*"
        }
    ]
}
```

   c. Scegli Review policy (Esamina policy).
   d. In Name (Nome), inserire **`IDTUsageMetricsIAMPermissions`**. In Riepilogo, esaminare le autorizzazioni concesse dai criteri.
   e. Scegli Create Policy (Crea policy).
3. Creare un utente IAM e assegnare le autorizzazioni agli utenti.

a. Creare un utente IAM. Segui i passaggi da 1 a 5 della sezione Creazione di utenti IAM (console) nella Guida per l'utente IAM. Se hai già creato un utente IAM, passa alla fase successiva.

b. Allega i comandi agli utenti IAM:

   i. Nella pagina Imposta autorizzazioni, seleziona Collega direttamente policy esistenti.

   ii. Cerca la policy IDTUsageMetrics IAmPermissions creata nella fase precedente. Selezionare la casella di controllo.

c. Scegli Successivo: Tag.

d. Scegliere Next:Review per visualizzare un riepilogo delle tue scelte.

e. Selezionare Create user (Crea utente).

f. Per visualizzare le chiavi di accesso dell'utente (ID chiave di accesso e chiavi di accesso segrete), scegliere Mostra accanto alla password e alla chiave di accesso. Per salvare le chiavi di accesso, scegliere Scarica .csv e salvare il file in una posizione sicura. Queste informazioni serviranno in seguito per configurare il file diAWS credenziali.

Per configurare le autorizzazioni per IDT (AWS CLI)

Attenersi alla seguente procedura per utilizzare l'AWS CLI per configurare le autorizzazioni per IDT per AWS IoT Greengrass.

1. Sul computer, installare e configurare l'AWS CLI se non è già installata. Segui i passaggi descritti in Installazione delAWS CLI manualeAWS Command Line Interface utente.

   > **Note**
   >
   > AWS CLIè uno strumento open source che è possibile utilizzare per interagire conAWS i servizi utilizzando i comandi nella shell a riga di comando.

2. Crea la seguente politica gestita dai clienti che concede le autorizzazioni per gestire IDT eAWS IoT Greengrass ruoli.

Linux or Unix

```
aws iam create-policy --policy-name IDTUsageMetricsIAMPermissions --policy-
document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot-device-tester:SendMetrics"
            ],
            "Resource": "*"
        }
    ]
}'
```

Windows command prompt

```
aws iam create-policy --policy-name IDTUsageMetricsIAMPermissions --policy-
document
                                          '{\"Version\": \"2012-10-17\",
 \"Statement\": [{\"Effect\": \"Allow\", \"Action\": [\"iot-device-
tester:SendMetrics\"], \"Resource\": \"*\"}]}'
```

> **Note**
>
> Questo passaggio include un esempio del prompt dei comandi di Windows perché utilizza una sintassi JSON diversa rispetto ai comandi del terminale Linux, macOS o Unix.

PowerShell

```
aws iam create-policy --policy-name IDTUsageMetricsIAMPermissions --policy-
document '{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "iot-device-tester:SendMetrics"
            ],
            "Resource": "*"
        }
    ]
}'
```

3. Creare un utente IAM e allegare i comandi richiesti da IDT perAWS IoT Greengrass.

   a. Creare un utente IAM.

   ```
   aws iam create-user --user-name user-name
   ```

   b. Allega la`IDTUsageMetricsIAMPermissions` policy che hai creato al tuo utente IAM. Sostituisci *il nome utente* con il tuo nome utente IAM e <account-id>nel comando con l'ID del tuoAccount AWS.

   ```
   aws iam attach-user-policy --user-name user-name --policy-arn
    arn:aws:iam::<account-id>:policy/IDTGreengrassIAMPermissions
   ```

4. Creare una chiave di accesso segreta per l'utente.

   ```
   aws iam create-access-key --user-name user-name
   ```

   Memorizzare l'output in una posizione sicura. Queste informazioni serviranno in seguito per configurare il file diAWS credenziali.

FornisciAWS credenziali a IDT

Per consentire a IDT di accedere alle tueAWS credenziali e inviare le metricheAWS, procedi come segue:

1. Memorizza leAWS credenziali per il tuo utente IAM come variabili di ambiente o in un file di credenziali:

   a. Per utilizzare le variabili di ambiente, esegui i seguenti comandi.

   Linux or Unix

   ```
   export AWS_ACCESS_KEY_ID=access-key
   ```

```
export AWS_SECRET_ACCESS_KEY=secret-access-key
```

Windows Command Prompt (CMD)

```
set AWS_ACCESS_KEY_ID=access-key
set AWS_SECRET_ACCESS_KEY=secret-access-key
```

PowerShell

```
$env:AWS_ACCESS_KEY_ID="access-key"
$env:AWS_SECRET_ACCESS_KEY="secret-access-key"
```

b. Per utilizzare il file delle credenziali, aggiungete le seguenti informazioni al`~/.aws/credentials` file.

```
[profile-name]
aws_access_key_id=access-key
aws_secret_access_key=secret-access-key
```

2. Configura la`auth` sezione del`config.json` file. Per ulteriori informazioni, consulta (Facoltativo) Configura config.json.

Risoluzione dei problemi IDT perAWS IoT GreengrassV2

IDT perAWS IoT GreengrassV2 scrive gli errori in varie posizioni in base al tipo di errore. IDT scrive gli errori nella console, nei file di registro e nei rapporti di test.

Dove cercare gli errori

Gli errori di alto livello vengono visualizzati sulla console durante l'esecuzione del test e, una volta completati tutti i test, viene visualizzato un riepilogo dei test non riusciti.`awsiotdevicetester_report.xml`contiene un riepilogo di tutti gli errori che hanno causato il fallimento di un test. IDT archivia i file di registro per ogni esecuzione del test in una directory con un UUID per l'esecuzione del test, visualizzato sulla console durante l'esecuzione del test.

La directory dei registri di test IDT è`<device-tester-extract-location>/results/<execution-id>/logs/`. Questa directory contiene i seguenti file visualizzati nella tabella. Questo parametro è utile per il debugging.

File	Descrizione
`test_manager.log`	I log scritti sulla console durante l'esecuzione del test. Il riepilogo dei risultati alla fine di questo file include un elenco dei test non riusciti. I log di avviso e di errore in questo file possono fornire informazioni sull'errore.
`test-group-id` `/`*`test-case-id`* `/`*`test-name`* `.log`	Registri dettagliati per il test specifico in un gruppo di test. Per i test che utilizzano componenti Greengrass, viene chiamato il file di registro del test case`greengrass-test-run.log` .
`test-group-id` `/`*`test-case-id`* `/greengrass.log`	Registri dettagliati perAWS IoT Greengrass Software di base. IDT copia questo file dal dispositivo sottoposto a test quando esegue i test di installazioneAWS IoT Greengrass Software di base sul dispositivo. Per ulteriori informazioni sui messaggi contenuti in questo file di registro, vedereRisoluzione dei problemi AWS IoT Greengrass V2.
`test-group-id` `/`*`test-case-id`*`/`*`component-name`* `.log`	Registri dettagliati per i componenti Greengrass che vengono distribuiti durante le esecuzioni di test. IDT copia i file di registro dei componenti dal dispositivo sottoposto a test quando esegue test che distribuiscono componenti specifici. Il nome di ogni file di registro dei componenti corrisponde al nome del componente distribuito. Per ulteriori informazioni sui messaggi in questo file di registro, vedereRisoluzione dei problemi AWS IoT Greengrass V2.

Risoluzione di IDT perAWS IoT GreengrassErrori V2

Prima di eseguire IDT perAWS IoT Greengrass, installa i file di configurazione corretti. Se si verificano errori di analisi e configurazione, il primo passaggio consiste nell'individuare e utilizzare un modello di configurazione appropriato per l'ambiente in uso.

Se continui a riscontare problemi, consulta la seguente procedura di debug.

Argomenti

errori di risoluzione degli alias

Quando esegui suite di test personalizzate, potresti visualizzare il seguente errore nella console e nel`test_manager.log`.

```
Couldn't resolve placeholders: couldn't do a json lookup: index out of range
```

Questo errore può verificarsi quando gli alias configurati nell'orchestrator di test IDT non si risolvono correttamente o se i valori risolti non sono presenti nei file di configurazione. Per risolvere questo errore, assicurati che`device.json`e`userdata.json` contengono le informazioni corrette richieste per la tua suite di test. Per informazioni sulla configurazione richiesta perAWS IoT Greengrassqualifica, vedereConfigurare le impostazioni IDT per eseguire la suite di AWS IoT Greengrass qualifiche.

Errori di conflitto

È possibile che venga visualizzato il seguente errore quando si esegue ilAWS IoT Greengrasssuite di qualificazione contemporaneamente su più di un dispositivo.

```
ConflictException: Component [com.example.IDTHelloWorld : 1.0.0] for account [account-
id] already exists with state: [DEPLOYABLE] { RespMetadata: { StatusCode: 409,
 RequestID: “id” }, Message_: “Component [com.example.IDTHelloWorld : 1.0.0] for
 account [account-id] already exists with state: [DEPLOYABLE]” }
```

L'esecuzione simultanea di test non è ancora supportata perAWS IoT Greengrasssuite di qualificazione. Esegui la suite di qualificazione in sequenza per ogni dispositivo.

Errore di avvio del test non riuscito

È possibile che si verifichino errori che indicano errori verificatisi durante il tentativo di avvio del test. Le cause sono diverse, quindi esegui le operazioni descritte di seguito:

- Assicurati che il nome del pool nel comando di esecuzione esista effettivamente. IDT fa riferimento al nome del pool direttamente dal`device.json`file.
- Verifica che i parametri di configurazione dei dispositivi nel pool siano corretti.

L'immagine di qualificazione Docker presenta errori

I test di qualificazione del gestore di applicazioni Docker utilizzano il`amazon/amazon-ec2-metadata-mock`immagine del contenitore in Amazon ECR per qualificare il dispositivo sottoposto a test.

Potresti ricevere il seguente errore se l'immagine è già presente in un contenitore Docker sul dispositivo in prova.

```
The Docker image amazon/amazon-ec2-metadata-mock:version already exists on the device.
```

Se in precedenza hai scaricato questa immagine ed eseguito il`amazon/amazon-ec2-metadata-mock`contenitore sul tuo dispositivo, assicurati di rimuovere questa immagine dal dispositivo sottoposto a test prima di eseguire i test di qualificazione.

Impossibile leggere le credenziali

Durante il test dei dispositivi Windows, potresti riscontrare`Failed to read credential`errore nel`greengrass.log`file se l'utente che usi per connetterti al dispositivo in prova non è configurato nel gestore delle credenziali su quel dispositivo.

Per risolvere questo errore, configura l'utente e la password per l'utente IDT nel gestore delle credenziali sul dispositivo in prova.

Per ulteriori informazioni, consulta Configura le credenziali utente per i dispositivi Windows.

Guidare gli errori con PreInstalled Greengrass

Durante l'esecuzione di IDT con PreInstalled Greengrass, se riscontri un errore di`GuiceoErrorInCustomProvider`, controlla se il file`userdata.json`ha il`InstalledDirRootOnDevice`impostato nella cartella di installazione di Greengrass. IDT verifica la presenza del file`effectiveConfig.yaml`sotto`<InstallationDirRootOnDevice>/config/effectiveConfig.yaml`.

Per ulteriori informazioni, consulta Configura le credenziali utente per i dispositivi Windows.

Eccezione di firma non valida

Quando esegui i test di qualificazione Lambda, potresti riscontrare il`invalidsignatureexception`errore se la macchina host IDT presenta problemi di accesso alla rete. Reimposta il router ed esegui nuovamente i test.

Errori di qualificazione dell'apprendimento automatico

Quando esegui test di qualificazione dell'apprendimento automatico (ML), potresti riscontrare errori di qualificazione se il tuo dispositivo non soddisfa irequisitiper implementare ilAWS- componenti ML forniti. Per risolvere gli errori di qualificazione ML, procedi come segue:

- Cerca i dettagli degli errori nei registri dei componenti che sono stati distribuiti durante l'esecuzione del test. I registri dei componenti si trovano nel`<device-tester-extract-location>/results/<execution-id>/logs/<test-group-id>`rubrica.

- Aggiungi il`-Dgg.persist=installed.software`argomento al`test.json`file per il test case fallito. `Latest.json`il file si trova in*`<device-tester-extract-location>`*`/tests/GGV2Q_`*`version`*` directory.`

Implementazioni di Open Test Framework (OTF) non riuscite

Se i test OTF non riescono a completare la distribuzione, una causa probabile potrebbe essere rappresentata dalle autorizzazioni impostate per la cartella principale di`TempResourcesDirOnDevice`e`InstallationDirRootOnDevice`. Per impostare correttamente le autorizzazioni di questa cartella, esegui il comando seguente. Sostituisci*`folder-name`*con il nome della cartella principale.

```
sudo chmod 755 folder-name
```

Errori di parsing

Gli errori di battitura in una configurazione JSON possono causare errori di analisi. Nella maggior parte dei casi, il problema è dovuto all'omissione di parentesi, virgole o virgolette nel file JSON. IDT esegue una convalida JSON e visualizza le informazioni di debug. Inoltre indica la riga in cui si è verificato l'errore, il numero di riga e il numero di colonna dell'errore di sintassi. Queste informazioni dovrebbero essere sufficienti per aiutarti a correggere l'errore, ma se ancora non riesci a individuare l'errore, puoi eseguire la convalida manualmente nel tuo IDE, in un editor di testo come Atom o Sublime o tramite uno strumento online come JSONLint.

Errori di autorizzazione negata

IDT esegue operazioni su varie directory e file in un dispositivo sottoposto a test. Alcune di queste operazioni richiedono l'accesso root. Per automatizzare queste operazioni, IDT deve essere in grado di eseguire comandi con il comando sudo senza la digitazione di una password.

Segui questi passaggi per consentire l'accesso al comando sudo senza la digitazione di una password.

> **Note**
>
> `user`e`username`si riferiscono all'utente SSH utilizzato da IDT per accedere al dispositivo sottoposto a test.

1. Utilizza sudo usermod -aG sudo ***<ssh-username>*** per aggiungere l'utente SSH al gruppo sudo
2. Per rendere effettive le modifiche, esci ed esegui di nuovo l'accesso.
3. Apri il file `/etc/sudoers` e aggiungi la riga seguente alla fine del file: *<ssh-username>* `ALL=(ALL) NOPASSWD: ALL`

> **Note**
>
> Come best practice, ti consigliamo di utilizzare sudo visudo quando modifichi `/etc/sudoers`.

Errore di generazione del rapporto di qualificazione

IDT supporta le quattro versioni più recenti*major*.*minor*versioni diAWS IoT Greengrasssuite di qualificazione V2 (GGV2Q) per generare report sulle qualifiche da inviare aAWS Partner Networkper includere i tuoi dispositivi nelAWS PartnerCatalogo dei dispositivi. Le versioni precedenti della suite di qualificazione non generano report sulle qualifiche.

Se hai domande sulla politica di supporto, contattaAWS Support.

Errore di parametro richiesto mancante

Quando IDT aggiunge nuove funzionalità, potrebbe apportare modifiche ai file di configurazione. L'utilizzo di un file di configurazione precedente potrebbe invalidare la tua configurazione. Se dovesse succedere, il file *<test_case_id>*`.log` in `/results/`*<execution-id>*`/logs` elenca in modo esplicito tutti i parametri mancanti. IDT convalida anche gli schemi dei file di configurazione JSON per verificare che stiate utilizzando l'ultima versione supportata.

Eccezione di sicurezza su macOS

Quando esegui IDT su un computer host macOS, ne blocca l'esecuzione. Per eseguire IDT, concedi un'eccezione di sicurezza agli eseguibili che fa parte della funzionalità di runtime IDT. Quando viene visualizzato il messaggio di avviso sul computer host, effettuate le seguenti operazioni per ciascuno degli eseguibili applicabili:

Per concedere un'eccezione di sicurezza agli eseguibili IDT

1. Sul computer macOS, nel menu Apple, apriPreferenze di sistema.

2. ScegliSicurezza e privacy, quindi suGeneralescheda, scegli l'icona del lucchetto per apportare modifiche alle impostazioni di sicurezza.
3. In caso di bloccodevicetester_mac_x86-64, cerca il messaggio`"devicetester_mac_x86-64" was blocked from use because it is not from an identified developer.`e scegliConsenti comunque.
4. Riprendi i test IDT, finché non avrai completato tutti gli eseguibili coinvolti.

Errori di connessione SSH

Quando IDT non riesce a connettersi a un dispositivo sottoposto a test, registra gli errori di connessione`/results/`*`<execution-id>`*`/logs/`*`<test-case-id>`*`.log`. I messaggi SSH vengono visualizzati nella parte superiore di questo file di registro perché la connessione a un dispositivo sottoposto a test è una delle prime operazioni eseguite da IDT.

La maggior parte delle configurazioni Windows utilizza l'applicazione terminale PuTTY per connettersi agli host Linux. Questa applicazione richiede la conversione dei file di chiave privata PEM standard in un formato Windows proprietario chiamato PPK. Se configuri SSH nel tuo`device.json`file, usa file PEM. Se si utilizza un file PPK, IDT non è in grado di creare una connessione SSH conAWS IoT Greengrassdispositivo e non può eseguire test.

A partire da IDT v4.4.0, se non hai abilitato SFTP sul dispositivo in prova, potresti vedere il seguente errore nel file di registro.

```
SSH connection failed with EOF
```

Per risolvere questo errore, abilita SFTP sul tuo dispositivo.

Errori di qualificazione dello Stream Manager

Quando esegui i test di qualificazione dello stream manager, potresti visualizzare il seguente errore nel`com.aws.StreamManagerExport.log`file.

```
Failed to upload data to S3
```

Questo errore può verificarsi quando lo stream manager utilizzaAWScredenziali in`~/root/.aws/credentials`file sul dispositivo invece di utilizzare le credenziali di ambiente che IDT esporta sul dispositivo in prova. Per evitare questo problema, elimina il`credentials`archivia il file sul dispositivo ed esegui nuovamente il test di qualificazione.

Errori di timeout

Puoi aumentare il timeout per ogni test specificando un moltiplicatore di timeout applicato al valore predefinito del timeout di ogni test. Qualsiasi valore configurato per questo flag deve essere maggiore o uguale a 1.0.

Per usare il moltiplicatore di timeout, utilizza il flag `--timeout-multiplier` durante l'esecuzione dei test. Ad esempio:

```
./devicetester_linux run-suite --suite-id GGV2Q_1.0.0 --pool-id DevicePool1 --timeout-multiplier 2.5
```

Per ulteriori informazioni, esegui `run-suite --help`.

Alcuni errori di timeout si verificano quando i test case IDT non possono essere completati a causa di problemi di configurazione. Non è possibile risolvere questi errori aumentando il moltiplicatore di timeout. Utilizza i log dell'esecuzione del test per risolvere i problemi di configurazione sottostanti.

- Se i log dei componenti MQTT o Lambda contengonoAccess deniederrori, la cartella di installazione di Greengrass potrebbe non avere le autorizzazioni di file corrette. Eseguite il seguente comando per ogni cartella nel percorso di installazione definito neluserdata.jsonfile.

  ```
  sudo chmod 755 folder-name
  ```

- Se i log di Greengrass indicano che l'implementazione della CLI di Greengrass non è completa, procedi come segue:
  - Verificate chebashè installato sul dispositivo in prova.
  - Se il tuouserdata.jsonil file includeGreengrassCliVersionparametro di configurazione, rimuoverlo. Questo parametro è obsoleto in IDT v4.1.0 e versioni successive. Per ulteriori informazioni, consulta Configura userdata.json.
- Se il test di implementazione Lambda ha esito negativo e viene visualizzato il messaggio di errore «Validating Lambda publish: timeout» e viene visualizzato un errore nel file di registro del test (`idt-gg2-lambda-function-idt-<resource-id>.log`) che dice`Error: Could not find or load main class com.amazonaws.greengrass.runtime.LambdaRuntime.`, procedi come segue:
  - Verifica per quale cartella è stata utilizzata`InstallationDirRootOnDeviceneluserdata.json`fascicolo.

- Assicurati che sul tuo dispositivo siano configurate le autorizzazioni utente corrette. Per ulteriori dettagli, consultaConfigura le autorizzazioni utente sul tuo dispositivo.

Errori di controllo della versione

IDT emette il seguente errore quandoAWSle credenziali utente per l'utente IDT non dispongono delle autorizzazioni IAM richieste.

```
Failed to check version compatibility
```

IlAWSutente che non dispone delle autorizzazioni IAM richieste.

Politica di supporto AWS IoT Device Tester per AWS IoT Greengrass

AWS IoT Device Tester for AWS IoT Greengrass è uno strumento di automazione dei test utilizzato per convalidare e qualificare i AWS IoT Greengrass dispositivi per l'inclusione nel Device Catalog.AWS Partner Ti consigliamo di utilizzare la versione più recente di AWS IoT Greengrass e AWS IoT Device Tester per testare o qualificare i tuoi dispositivi.

AWS IoT Device Tester È disponibile almeno una versione di per ogni versione supportata di AWS IoT Greengrass. Per le versioni supportate di AWS IoT Greengrass, vedere Versioni Greengrass nucleus. Per le versioni supportate di AWS IoT Device Tester, vedere. Versioni supportate di AWS IoT Device Tester for AWS IoT Greengrass V2

Puoi anche utilizzare una qualsiasi delle versioni supportate di AWS IoT Greengrass e AWS IoT Device Tester per testare o qualificare i tuoi dispositivi. Sebbene sia possibile continuare a utilizzare versioni non supportate di AWS IoT Device Tester, tali versioni non ricevono correzioni di bug o aggiornamenti. Se hai domande sulla politica di supporto, contatta. AWS Support

Soluzioni IoT basate su Greengrass

Everyware di Eurotech GreenEdge è disponibile in anteprima ed è soggetto a modifiche. AWS IoT Greengrass Questa soluzione non è supportata da AWS. È necessario contattare Eurotech per qualsiasi problema con questo dispositivo.

AWS IoT Greengrassoffre soluzioni dei partner per ottimizzare la tua esperienza di installazione di Greengrass. Di seguito è riportata una soluzione AWS offerta in collaborazione con Eurotech. Questa soluzione include AWS IoT Greengrass Core Edge Runtime e funzionalità aggiuntive preinstallate.

Eurotech

AWSha collaborato con Eurotech per offrire una soluzione IoT per i clienti che cercano un dispositivo dotato del software AWS IoT Greengrass Core preinstallato. Everyware di Eurotech GreenEdge è un software IoT edge preconfigurato e prequalificato da. AWS Questa soluzione unisce le funzionalità di Greengrass e di Eurotech Everyware Software Framework (ESF) per offrire ai clienti un'ampia connettività in direzione sud tramite adattatori di protocollo come: Modbus, OPC-UA Client/Server, S7, TwinCat, J1939, DNP3 Master/Outstation e altri. Con questa soluzione, puoi anche inviare dati a Cloud AWS e connetterti a tutti i AWS servizi in direzione nord (comeAWS IoT Core, AWS IoT SiteWiseAWS IoT Analytics, Amazon S3 e Amazon Kinesis Video Streams). In combinazione con Everyware Cloud, la soluzione di gestione dei dispositivi di Eurotech, questa soluzione introduce un nuovo servizio Zero-Touch Provisioning, che semplifica l'onboarding e l'implementazione di massa dei dispositivi.

Per ulteriori informazioni su Eurotech, vedere Eurotech.

Risoluzione dei problemi AWS IoT Greengrass V2

Utilizza le informazioni e le soluzioni per la risoluzione dei problemi contenute in questa sezione per risolvere i problemi relativi a. AWS IoT Greengrass Version 2

Argomenti

Visualizza i registri del software e dei componenti AWS IoT Greengrass principali

Il software AWS IoT Greengrass Core scrive i log nel file system locale che è possibile utilizzare per visualizzare informazioni in tempo reale sul dispositivo principale. Puoi anche configurare i dispositivi principali per scrivere i log nei CloudWatch registri, in modo da poter risolvere i problemi dei dispositivi principali in remoto. Questi registri possono aiutarti a identificare i problemi relativi ai componenti, alle implementazioni e ai dispositivi principali. Per ulteriori informazioni, consulta Monitora AWS IoT Greengrass i registri.

AWS IoT Greengrass Problemi software principali

Risolvi i problemi relativi AWS IoT Greengrass al software Core.

Argomenti

- Impossibile configurare il dispositivo principale
- Impossibile avviare il software AWS IoT Greengrass Core come servizio di sistema
- Impossibile configurare nucleus come servizio di sistema
- Impossibile connettersi a AWS IoT Core
- Errore di memoria esaurita
- Impossibile installare Greengrass CLI
- User root is not allowed to execute
- com.aws.greengrass.lifecyclemanager.GenericExternalService: Could not determine user/group to run with
- Failed to map segment from shared object: operation not permitted
- Impossibile configurare il servizio Windows
- com.aws.greengrass.util.exceptions.TLSAuthException: Failed to get trust manager
- com.aws.greengrass.deployment.IotJobsHelper: No connection available during subscribing to Iot Jobs descriptions topic. Will retry in sometime
- software.amazon.awssdk.services.iam.model.IamException: The security token included in the request is invalid
- software.amazon.awssdk.services.iot.model.IotException: User: <user> is not authorized to perform: iot:GetPolicy
- Error: com.aws.greengrass.shadowmanager.sync.model.FullShadowSyncRequest: Could not execute cloud shadow get request
- Operation aws.greengrass#<operation> is not supported by Greengrass
- java.io.FileNotFoundException: <stream-manager-store-root-dir>/stream_manager_metadata_store (Permission denied)
- com.aws.greengrass.security.provider.pkcs11.PKCS11CryptoKeyService: Private key or certificate with label <label> does not exist
- software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException: User: <user> is not authorized to perform: secretsmanager:GetSecretValue on resource: <arn>
- software.amazon.awssdk.services.secretsmanager.model.SecretsManagerException: Access to KMS is not allowed
- java.lang.NoClassDefFoundError: com/aws/greengrass/security/CryptoKeySpi
- com.aws.greengrass.security.provider.pkcs11.PKCS11CryptoKeyService: CKR_OPERATION_NOT_INITIALIZED

- Greengrass core device stuck on nucleus v2.12.3

Impossibile configurare il dispositivo principale

Se il programma di installazione del software AWS IoT Greengrass Core fallisce e non riesci a configurare un dispositivo principale, potresti dover disinstallare il software e riprovare. Per ulteriori informazioni, consulta Disinstalla il software AWS IoT Greengrass Core.

Impossibile avviare il software AWS IoT Greengrass Core come servizio di sistema

Se il software AWS IoT Greengrass Core non si avvia, controlla i registri dei servizi di sistema per identificare il problema. Un problema comune è rappresentato dalla mancata disponibilità di Java nella variabile di ambiente PATH (Linux) o nella variabile di sistema PATH (Windows).

Impossibile configurare nucleus come servizio di sistema

Potresti visualizzare questo errore quando il programma di installazione del software AWS IoT Greengrass Core non riesce a configurarsi AWS IoT Greengrass come servizio di sistema. Sui dispositivi Linux, questo errore si verifica in genere se il dispositivo principale non dispone del sistema systemd init. Il programma di installazione può configurare correttamente il software AWS IoT Greengrass Core anche se non riesce a configurare il servizio di sistema.

Esegui una di queste operazioni:

- Configura ed esegui il software AWS IoT Greengrass Core come servizio di sistema. È necessario configurare il software come servizio di sistema per utilizzare tutte le funzionalità di AWS IoT Greengrass. È possibile installare systemd o utilizzare un sistema di inizializzazione diverso. Per ulteriori informazioni, consulta Configurare il nucleo Greengrass come servizio di sistema.
- Esegui il software AWS IoT Greengrass Core senza un servizio di sistema. È possibile eseguire il software utilizzando uno script di caricamento che il programma di installazione configura nella cartella principale di Greengrass. Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core senza un servizio di sistema.

Impossibile connettersi a AWS IoT Core

Potresti visualizzare questo errore quando il software AWS IoT Greengrass Core non riesce a connettersi AWS IoT Core a per recuperare i lavori di distribuzione, ad esempio. Esegui questa operazione:

- Verifica che il tuo dispositivo principale sia in grado di connettersi a Internet e AWS IoT Core. Per ulteriori informazioni sull' AWS IoT Core endpoint a cui si connette il dispositivo, consultaConfigurare il software AWS IoT Greengrass Core.
- Verifica che il dispositivo AWS IoT principale del tuo dispositivo utilizzi un certificato che consenta le`iot:Connect`, `iot:Publish`iot:Receive, e le `iot:Subscribe` autorizzazioni.
- Se il dispositivo principale utilizza un proxy di rete, verifica che il dispositivo principale abbia un ruolo di dispositivo e che tale ruolo consenta le`iot:Connect`, `iot:Publish`iot:Receive, e le `iot:Subscribe` autorizzazioni.

Errore di memoria esaurita

Questo errore si verifica in genere se il dispositivo non dispone di memoria sufficiente per allocare un oggetto nell'heap Java. Sui dispositivi con memoria limitata, potrebbe essere necessario specificare una dimensione massima dell'heap per controllare l'allocazione della memoria. Per ulteriori informazioni, consulta Controlla l'allocazione della memoria con le opzioni JVM.

Impossibile installare Greengrass CLI

È possibile che venga visualizzato il seguente messaggio sulla console quando si utilizza l'`--deploy-dev-tools`argomento nel comando di installazione per AWS IoT Greengrass Core.

```
Thing group exists, it could have existing deployment and devices, hence NOT creating
 deployment for Greengrass first party dev tools, please manually create a deployment
 if you wish to
```

Ciò si verifica quando il componente Greengrass CLI non è installato perché il dispositivo principale è membro di un gruppo di oggetti con una distribuzione esistente. Se vedi questo messaggio, puoi distribuire manualmente il `aws.greengrass.Cli` componente Greengrass CLI () sul dispositivo per installare la Greengrass CLI. Per ulteriori informazioni, consulta Installazione della CLI di Greengrass.

User root is not allowed to execute

Potresti visualizzare questo errore quando l'utente che esegue il software AWS IoT Greengrass Core (in genere`root`) non dispone dell'autorizzazione per l'esecuzione `sudo` con alcun utente e gruppo. Per l'utente `ggc_user` di sistema predefinito, questo errore è simile al seguente:

```
Sorry, user root is not allowed to execute <command> as ggc_user:ggc_group.
```

Verifica che il `/etc/sudoers` file dia all'utente il permesso di funzionare `sudo` come altri gruppi. L'autorizzazione per l'utente `/etc/sudoers` dovrebbe essere simile all'esempio seguente.

```
root    ALL=(ALL:ALL) ALL
```

com.aws.greengrass.lifecyclemanager.GenericExternalService: Could not determine user/group to run with

Potresti visualizzare questo errore quando il dispositivo principale tenta di eseguire un componente e il nucleo di Greengrass non specifica un utente di sistema predefinito da utilizzare per eseguire i componenti.

Per risolvere questo problema, configura il nucleo Greengrass per specificare l'utente di sistema predefinito che esegue i componenti. Per ulteriori informazioni, consulta Configurare l'utente che esegue i componenti e Configura l'utente predefinito del componente.

Failed to map segment from shared object: operation not permitted

Potresti visualizzare questo errore quando il software AWS IoT Greengrass Core non si avvia perché la `/tmp` cartella è montata con `noexec` le autorizzazioni. La libreria AWS Common Runtime (CRT) utilizza la `/tmp` cartella per impostazione predefinita.

Esegui una di queste operazioni:

- Esegui il comando seguente per rimontare la `/tmp` cartella con le `exec` autorizzazioni e riprova.

  ```
  sudo mount -o remount,exec /tmp
  ```

- Se si esegue Greengrass nucleus v2.5.0 o versione successiva, è possibile impostare un'opzione JVM per modificare la cartella utilizzata dalla libreria CRT. AWS È possibile specificare il

`jvmOptions` parametro nella configurazione del componente Greengrass nucleus in una distribuzione o quando si installa il AWS IoT Greengrass software Core. Sostituisci */path/to/use* con il percorso di una cartella utilizzabile dalla libreria CRT. AWS

```
{
  "jvmOptions": "-Daws.crt.lib.dir=\"/path/to/use\""
}
```

Impossibile configurare il servizio Windows

Potresti visualizzare questo errore se installi il software AWS IoT Greengrass Core su un dispositivo Microsoft Windows 2016. Il software AWS IoT Greengrass Core non è supportato in Windows 2016. Per un elenco dei sistemi operativi supportati, consultaPiattaforme supportate.

Se è necessario utilizzare Windows 2016, è possibile effettuare le seguenti operazioni:

1. Decomprimi l'archivio di installazione AWS IoT Greengrass Core scaricato
2. Nella `Greengrass` directory apri il `bin/greengrass.xml.template` file.
3. Aggiungi il `<autoRefresh>` tag alla fine del file appena prima del `</service>` tag.

```
  </log>
  <autoRefresh>false</autoRefresh>
</service>
```

com.aws.greengrass.util.exceptions.TLSAuthException: Failed to get trust manager

Potresti visualizzare questo errore quando installi il software AWS IoT Greengrass Core senza un file di autorità di certificazione (CA) principale.

```
2022-06-05T10:00:39.556Z [INFO] (main) com.aws.greengrass.lifecyclemanager.Kernel:
 service-loaded. {serviceName=DeploymentService}
2022-06-05T10:00:39.943Z [WARN] (main)
 com.aws.greengrass.componentmanager.ClientConfigurationUtils: configure-greengrass-
mutual-auth. Error during configure greengrass client mutual auth. {}
com.aws.greengrass.util.exceptions.TLSAuthException: Failed to get trust manager
```

Verificate di aver specificato un file CA root valido con il `rootCaPath` parametro nel file di configurazione fornito all'installatore. Per ulteriori informazioni, consulta Installare il software AWS IoT Greengrass Core..

com.aws.greengrass.deployment.IotJobsHelper: No connection available during subscribing to Iot Jobs descriptions topic. Will retry in sometime

Potresti visualizzare questo messaggio di avviso quando il dispositivo principale non riesce a connettersi per iscriversi AWS IoT Core alle notifiche dei lavori di distribuzione. Esegui questa operazione:

- Verifica che il dispositivo principale sia connesso a Internet e possa raggiungere l'endpoint di AWS IoT dati che hai configurato. Per ulteriori informazioni sugli endpoint utilizzati dai dispositivi principali, consulta. Consenti il traffico dei dispositivi tramite un proxy o un firewall
- Controlla i log di Greengrass per altri errori che rivelano altre cause principali.

software.amazon.awssdk.services.iam.model.IamException: The security token included in the request is invalid

Potresti visualizzare questo errore quando installi il software AWS IoT Greengrass Core con provisioning automatico e il programma di installazione utilizza un token di AWS sessione non valido. Esegui questa operazione:

- Se utilizzate credenziali di sicurezza temporanee, verificate che il token di sessione sia corretto e che stiate copiando e incollando il token di sessione completo.
- Se utilizzi credenziali di sicurezza a lungo termine, verifica che il dispositivo non disponga di un token di sessione risalente a quando in precedenza utilizzavi credenziali temporanee. Esegui questa operazione:
  1. Esegui il comando seguente per annullare l'impostazione della variabile di ambiente del token di sessione.

     Linux or Unix

     ```
     unset AWS_SESSION_TOKEN
     ```

Windows Command Prompt (CMD)

```
set AWS_SESSION_TOKEN=
```

PowerShell

```
Remove-Item Env:\AWS_SESSION_TOKEN
```

2. Controlla se il file AWS delle credenziali contiene un token di sessione`~/.aws/credentials`,. `aws_session_token` In tal caso, rimuovi quella riga dal file.

```
aws_session_token = AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4Olgk
```

Puoi anche installare il software AWS IoT Greengrass Core senza fornire AWS credenziali. Per ulteriori informazioni, consulta Installa il software AWS IoT Greengrass Core con provisioning manuale delle risorse o Installa il software AWS IoT Greengrass Core con il provisioning AWS IoT della flotta.

software.amazon.awssdk.services.iot.model.IotException: User: <user> is not authorized to perform: iot:GetPolicy

Potresti visualizzare questo errore quando installi il software AWS IoT Greengrass Core con provisioning automatico e il programma di installazione utilizza AWS credenziali che non dispongono delle autorizzazioni richieste. Per ulteriori informazioni sulle autorizzazioni richieste, consulta. Policy IAM minima per l'installatore per il provisioning delle risorse

Controlla le autorizzazioni per l'identità IAM delle credenziali e concedi all'identità IAM tutte le autorizzazioni richieste mancanti.

Error: com.aws.greengrass.shadowmanager.sync.model.FullShadowSyncRequest: Could not execute cloud shadow get request

Potresti visualizzare questo errore quando usi il componente shadow manager per sincronizzare le ombre del dispositivo con. AWS IoT Core Il codice di stato HTTP 403 indica che questo errore si

è verificato perché la AWS IoT politica del dispositivo principale non concede l'autorizzazione alla chiamata. `GetThingShadow`

```
com.aws.greengrass.shadowmanager.sync.model.FullShadowSyncRequest: Could not execute
 cloud shadow get request. {thing name=MyGreengrassCore, shadow name=MyShadow}
2021-07-14T21:09:02.456Z [ERROR] (pool-2-thread-109)
 com.aws.greengrass.shadowmanager.sync.SyncHandler: sync. Skipping sync request. {thing
 name=MyGreengrassCore, shadow name=MyShadow}
com.aws.greengrass.shadowmanager.exception.SkipSyncRequestException:
 software.amazon.awssdk.services.iotdataplane.model.IotDataPlaneException:
 null (Service: IotDataPlane, Status Code: 403, Request ID:
 f6e713ba-1b01-414c-7b78-5beb3f3ad8f6, Extended Request ID: null)
```

Per sincronizzare le ombre locali con AWS IoT Core, la AWS IoT politica del dispositivo principale deve concedere le seguenti autorizzazioni:

- `iot:GetThingShadow`
- `iot:UpdateThingShadow`
- `iot:DeleteThingShadow`

Controlla la AWS IoT politica del dispositivo principale e aggiungi le autorizzazioni necessarie mancanti. Per ulteriori informazioni, consulta gli argomenti seguenti:

- AWS IoT Core azioni politiche nella Guida per gli AWS IoT sviluppatori
- Aggiorna la politica di un dispositivo principale AWS IoT

Operation aws.greengrass#<operation> is not supported by Greengrass

Potresti visualizzare questo errore quando utilizzi un'operazione di comunicazione tra processi (IPC) in un componente Greengrass personalizzato e il componente AWS fornito richiesto non è installato sul dispositivo principale.

Per risolvere questo problema, aggiungi il componente richiesto come dipendenza nella ricetta del componente, in modo che il software AWS IoT Greengrass Core abbia installato il componente richiesto al momento della distribuzione del componente.

- Recupera valori segreti: `aws.greengrass.SecretManager`
- Interagisci con le ombre locali — `aws.greengrass.ShadowManager`

- Gestisci le distribuzioni e i componenti locali, versione 2.6.0 o successiva `aws.greengrass.Cli`
- Autentica e autorizza i dispositivi client: v2.2.0 o versione successiva `aws.greengrass.clientdevices.Auth`

java.io.FileNotFoundException: <stream-manager-store-root-dir>/stream_manager_metadata_store (Permission denied)

Potresti visualizzare questo errore nel file di log di stream manager (`aws.greengrass.StreamManager.log`) quando configuri stream manager per utilizzare una cartella principale che non esiste o dispone delle autorizzazioni corrette. Per ulteriori informazioni su come configurare questa cartella, consulta la configurazione dello stream manager.

com.aws.greengrass.security.provider.pkcs11.PKCS11CryptoKeyService: Private key or certificate with label <label> does not exist

Questo errore si verifica quando il componente del provider PKCS #11 non riesce a trovare o caricare la chiave privata o il certificato specificato quando si configura il software AWS IoT Greengrass Core per l'utilizzo di un modulo di sicurezza hardware (HSM). Esegui questa operazione:

- Verificate che la chiave privata e il certificato siano archiviati nell'HSM utilizzando lo slot, il PIN utente e l'etichetta dell'oggetto che configurate per l'uso del software AWS IoT Greengrass Core.
- Verificate che la chiave privata e il certificato utilizzino la stessa etichetta dell'oggetto nell'HSM.
- Se il tuo HSM supporta gli ID degli oggetti, verifica che la chiave privata e il certificato utilizzino lo stesso ID di oggetto nell'HSM.

Consulta la documentazione del tuo HSM per scoprire come richiedere dettagli sui token di sicurezza presenti nell'HSM. Se devi modificare lo slot, l'etichetta dell'oggetto o l'ID dell'oggetto per un token di sicurezza, consulta la documentazione del tuo HSM per scoprire come farlo.

software.amazon.awssdk.services.secretsmanager.model.SecretsManagerExcep User: <user> is not authorized to perform: secretsmanager:GetSecretValue on resource: <arn>

Questo errore può verificarsi quando si utilizza il componente Secret Manager per distribuire un AWS Secrets Manager segreto. Se il ruolo IAM di scambio di token del dispositivo principale non concede

l'autorizzazione per ottenere il segreto, l'implementazione fallisce e i log di Greengrass includono questo errore.

Per autorizzare un dispositivo principale a scaricare un segreto

1. Aggiungi l'`secretsmanager:GetSecretValue`autorizzazione al ruolo di scambio di token del dispositivo principale. La seguente dichiarazione politica di esempio concede il permesso di ottenere il valore di un segreto.

   ```
   {
       "Effect": "Allow",
       "Action": [
           "secretsmanager:GetSecretValue"
       ],
       "Resource": [
           "arn:aws:secretsmanager:us-west-2:123456789012:secret:MyGreengrassSecret-
   abcdef"
       ]
   }
   ```

   Per ulteriori informazioni, consulta Autorizza i dispositivi principali a interagire conAWSservizi.

2. Riapplica la distribuzione al dispositivo principale. Esegui una di queste operazioni:
   - Rivedi la distribuzione senza apportare modifiche. Il dispositivo principale tenta di scaricare nuovamente il segreto quando riceve la distribuzione modificata. Per ulteriori informazioni, consulta Rivedi le distribuzioni.
   - Riavvia il software AWS IoT Greengrass Core per riprovare la distribuzione. Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core

   La distribuzione ha esito positivo se il gestore segreto scarica il segreto con successo.

software.amazon.awssdk.services.secretsmanager.model.SecretsManagerExcep Access to KMS is not allowed

Questo errore può verificarsi quando si utilizza il componente del gestore segreto per distribuire un AWS Secrets Manager segreto crittografato da una AWS Key Management Service chiave. Se il ruolo IAM di scambio di token del dispositivo principale non concede l'autorizzazione per decrittografare il segreto, l'implementazione fallisce e i log di Greengrass includono questo errore.

Per risolvere il problema, aggiungi l'`kms:Decrypt`autorizzazione al ruolo di scambio di token del dispositivo principale. Per ulteriori informazioni, consulta gli argomenti seguenti:

- Crittografia e decrittografia segrete nella Guida per l'utente AWS Secrets Manager
- Autorizza i dispositivi principali a interagire conAWSservizi

java.lang.NoClassDefFoundError: com/aws/greengrass/security/CryptoKeySpi

Potresti visualizzare questo errore quando tenti di installare il software AWS IoT Greengrass Core con sicurezza hardware e utilizzi una versione precedente di Greengrass nucleus che non supporta l'integrazione della sicurezza hardware. Per utilizzare l'integrazione della sicurezza hardware, è necessario utilizzare Greengrass nucleus v2.5.3 o versione successiva.

com.aws.greengrass.security.provider.pkcs11.PKCS11CryptoKeyService: CKR_OPERATION_NOT_INITIALIZED

Potresti visualizzare questo errore quando usi la libreria TPM2 quando esegui Core come servizio di sistema. AWS IoT Greengrass

Questo errore indica che è necessario aggiungere una variabile di ambiente che fornisca la posizione dell'archivio PKCS #11 nel file di servizio AWS IoT Greengrass Core systemd.

Per ulteriori informazioni, consultate la sezione Requisiti della documentazione del Fornitore PKCS #11 componente.

Greengrass core device stuck on nucleus v2.12.3

Se il dispositivo principale Greengrass non modifica la distribuzione dalla versione Nucleus 2.12.3, potrebbe essere necessario scaricare e sostituire il file `Greengrass.jar` con Greengrass nucleus versione 2.12.2. Esegui questa operazione:

1. Sul tuo dispositivo Greengrass core, esegui il seguente comando per arrestare il software Greengrass Core.

   Linux or Unix

   ```
   sudo systemctl stop greengrass
   ```

Windows Command Prompt (CMD)

```
sc stop "greengrass"
```

PowerShell

```
Stop-Service -Name "greengrass"
```

2. Sul tuo dispositivo principale, scarica il AWS IoT Greengrass software in un file denominato. `greengrass-2.12.2.zip`

   Linux or Unix

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-2.12.2.zip >
    greengrass-2.12.2.zip
   ```

   Windows Command Prompt (CMD)

   ```
   curl -s https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-2.12.2.zip >
    greengrass-2.12.2.zip
   ```

   PowerShell

   ```
   iwr -Uri https://d2s8p88vqu9w66.cloudfront.net/releases/greengrass-2.12.2.zip -
   OutFile greengrass-2.12.2.zip
   ```

3. Decomprimi il software AWS IoT Greengrass Core in una cartella sul dispositivo. Sostituiscilo *GreengrassInstaller*con la cartella che desideri utilizzare.

   Linux or Unix

   ```
   unzip greengrass-2.12.2.zip -d GreengrassInstaller && rm greengrass-2.12.2.zip
   ```

   Windows Command Prompt (CMD)

   ```
   mkdir GreengrassInstaller && tar -xf greengrass-2.12.2.zip -
   C GreengrassInstaller && del greengrass-2.12.2.zip
   ```

PowerShell

```
Expand-Archive -Path greengrass-2.12.2.zip -DestinationPath .\
\GreengrassInstaller
rm greengrass-2.12.2.zip
```

4. Esegui il comando seguente per sovrascrivere il file JAR di nucleus versione 2.12.3 Greengrass con il file JAR Greengrass versione 2.12.2 di nucleus.

   Linux or Unix

   ```
   sudo cp ./GreengrassInstaller/lib/Greengrass.jar /greengrass/v2/packages/
   artifacts-unarchived/aws.greengrass.Nucleus/2.12.3/aws.greengrass.nucleus/lib
   ```

   Windows Command Prompt (CMD)

   ```
   robocopy ./GreengrassInstaller/lib/Greengrass.jar /greengrass/v2/packages/
   artifacts-unarchived/aws.greengrass.Nucleus/2.12.3/aws.greengrass.nucleus/lib /E
   ```

   PowerShell

   ```
   cp -Path ./GreengrassInstaller/lib/Greengrass.jar -Destination /
   greengrass/v2/packages/artifacts-unarchived/aws.greengrass.Nucleus/2.12.3/
   aws.greengrass.nucleus/lib
   ```

5. Esegui il seguente comando per avviare il software Greengrass Core.

   Linux or Unix

   ```
   sudo systemctl start greengrass
   ```

   Windows Command Prompt (CMD)

   ```
   sc start "greengrass"
   ```

   PowerShell

   ```
   Start-Service -Name "greengrass"
   ```

AWS IoT Greengrass problemi relativi al cloud

Utilizza le seguenti informazioni per risolvere i problemi relativi alla AWS IoT Greengrass console e all'API. Ogni voce corrisponde a un messaggio di errore che potresti visualizzare quando esegui un'azione.

An error occurred (AccessDeniedException) when calling the CreateComponentVersion operation: User: arn:aws:iam::123456789012:user/<username> is not authorized to perform: null

Potresti visualizzare questo errore quando crei una versione del componente dalla AWS IoT Greengrass console o con l'CreateComponentVersionoperazione.

Questo errore indica che la ricetta non è valida in formato JSON o YAML. Controlla la sintassi della ricetta, risolvi eventuali problemi di sintassi e riprova. Puoi utilizzare un correttore di sintassi JSON o YAML online per identificare i problemi di sintassi nella tua ricetta.

Invalid Input: Encountered following errors in Artifacts: {<s3ArtifactUri> = Specified artifact resource cannot be accessed}

Potresti visualizzare questo errore quando crei una versione del componente dalla console o con l' AWS IoT Greengrass operazione. CreateComponentVersion Questo errore indica che un artefatto S3 nella ricetta del componente non è valido.

Esegui questa operazione:

- Verifica che il bucket S3 si trovi nello stesso Regione AWS punto in cui crei il componente. AWS IoT Greengrass non supporta le richieste interregionali per gli artefatti dei componenti.
- Verifica che l'URI dell'artefatto sia un URL di oggetto S3 valido e verifica che l'artefatto esista nell'URL dell'oggetto S3.
- Verifica di disporre dell'autorizzazione per accedere all'artefatto all'URL dell'oggetto S3. Account AWS

INACTIVE deployment status

Potresti ottenere uno stato di `INACTIVE` implementazione quando chiami l'ListDeploymentsAPI senza le politiche dipendenti richieste. AWS IoT È necessario disporre delle autorizzazioni necessarie per ottenere uno stato di distribuzione accurato. Puoi trovare le azioni dipendenti cercando nelle Azioni definite da AWS IoT Greengrass V2 e seguendo le autorizzazioni necessarie per. `ListDeployments` Senza le AWS IoT autorizzazioni dipendenti richieste, continuerai a visualizzare lo stato della distribuzione, ma potresti visualizzare uno stato di distribuzione impreciso di. `INACTIVE`

Problemi principali di distribuzione dei dispositivi

Risolvi i problemi di distribuzione sui dispositivi core Greengrass. Ogni voce corrisponde a un messaggio di registro che potresti visualizzare sul tuo dispositivo principale.

Argomenti

- Error: com.aws.greengrass.componentmanager.exceptions.PackageDownloadException: Failed to download artifact
- Error: com.aws.greengrass.componentmanager.exceptions.ArtifactChecksumMismatchException: Integrity check for downloaded artifact failed. Probably due to file corruption.
- Error: com.aws.greengrass.componentmanager.exceptions.NoAvailableComponentVersionException: Failed to negotiate component <name> version with cloud and no local applicable version satisfying requirement <requirements>
- software.amazon.awssdk.services.greengrassv2data.model.ResourceNotFoundException: The latest version of Component <componentName> doesn't claim platform <coreDevicePlatform> compatibility
- com.aws.greengrass.componentmanager.exceptions.PackagingException: The deployment attempts to update the nucleus from aws.greengrass.Nucleus-<version> to aws.greengrass.Nucleus-<version> but no component of type nucleus was included as target component
- Error: com.aws.greengrass.deployment.exceptions.DeploymentException: Unable to process deployment. Greengrass launch directory is not set up or Greengrass is not set up as a system service
- Info: com.aws.greengrass.deployment.exceptions.RetryableDeploymentDocumentDownloadException: Greengrass Cloud Service returned an error when getting full deployment configuration

- Warn: com.aws.greengrass.deployment.DeploymentService: Failed to get thing group hierarchy
- Info: com.aws.greengrass.deployment.DeploymentDocumentDownloader: Calling Greengrass cloud to get full deployment configuration
- Caused by: software.amazon.awssdk.services.greengrassv2data.model.GreengrassV2DataException: null (Service: GreengrassV2Data, Status Code: 403, Request ID: <some_request_id>, Extended Request ID: null)

Error: com.aws.greengrass.componentmanager.exceptions.PackageDownloadExceptio Failed to download artifact

Potresti visualizzare questo errore quando il software AWS IoT Greengrass Core non riesce a scaricare un elemento del componente quando il dispositivo principale applica una distribuzione. La distribuzione non riesce a causa di questo errore.

Quando si riceve questo errore, il registro include anche una traccia dello stack che è possibile utilizzare per identificare il problema specifico. Ciascuna delle seguenti voci corrisponde a un messaggio che è possibile visualizzare nello stack trace del messaggio di `Failed to download artifact` errore.

Argomenti

- software.amazon.awssdk.services.s3.model.S3Exception: null (Service: S3, Status Code: 403, Request ID: null, ...)
- software.amazon.awssdk.services.s3.model.S3Exception: Access Denied (Service: S3, Status Code: 403, Request ID: <requestID>

software.amazon.awssdk.services.s3.model.S3Exception: null (Service: S3, Status Code: 403, Request ID: null, ...)

L'PackageDownloadException errore potrebbe includere questa traccia dello stack nei seguenti casi:

- L'elemento del componente non è disponibile all'URL dell'oggetto S3 specificato nella ricetta del componente. Verifica di aver caricato l'artefatto nel bucket S3 e che l'URI dell'artefatto corrisponda all'URL dell'oggetto S3 dell'artefatto nel bucket.

- Il ruolo di scambio di token del dispositivo principale non consente al software AWS IoT Greengrass Core di scaricare l'elemento del componente dall'URL dell'oggetto S3 specificato nella ricetta del componente. Verifica che il ruolo di scambio di token `s3:GetObject` consenta l'URL dell'oggetto S3 in cui è disponibile l'artefatto.

software.amazon.awssdk.services.s3.model.S3Exception: Access Denied (Service: S3, Status Code: 403, Request ID: <requestID>

L'PackageDownloadException errore potrebbe includere questa traccia dello stack quando il dispositivo principale non è autorizzato a chiamare. `s3:GetBucketLocation` Il messaggio di errore include anche il seguente messaggio.

```
reason: Failed to determine S3 bucket location
```

Verifica che il ruolo di scambio di token del dispositivo principale consenta `s3:GetBucketLocation` il bucket S3 in cui è disponibile l'artefatto.

Error: com.aws.greengrass.componentmanager.exceptions.ArtifactChecksumMismatchI Integrity check for downloaded artifact failed. Probably due to file corruption.

Potresti visualizzare questo errore quando il software AWS IoT Greengrass Core non riesce a scaricare un elemento del componente quando il dispositivo principale applica una distribuzione. La distribuzione non riesce perché il checksum del file di artefatto scaricato non corrisponde al checksum AWS IoT Greengrass calcolato al momento della creazione del componente.

Esegui questa operazione:

- Controlla se il file degli artefatti è cambiato nel bucket S3 in cui lo ospiti. Se il file è cambiato dopo la creazione del componente, ripristinalo alla versione precedente prevista dal dispositivo principale. Se non riesci a ripristinare il file alla versione precedente o se desideri utilizzare la nuova versione del file, crea una nuova versione del componente con il file dell'artefatto.
- Controlla la connessione Internet del tuo dispositivo principale. Questo errore può verificarsi se il file dell'artefatto viene danneggiato durante il download. Crea una nuova distribuzione e riprova.

Error: com.aws.greengrass.componentmanager.exceptions.NoAvailableComponentVers Failed to negotiate component <name> version with cloud and no local applicable version satisfying requirement <requirements>

Potresti visualizzare questo errore quando un dispositivo principale non riesce a trovare una versione del componente che soddisfi i requisiti delle distribuzioni per quel dispositivo principale. Il dispositivo principale verifica la presenza del componente nel AWS IoT Greengrass servizio e sul dispositivo locale. Il messaggio di errore include la destinazione di ogni distribuzione e i requisiti di versione di tale distribuzione per il componente. L'obiettivo di distribuzione può essere un oggetto, un gruppo di oggetti `oLOCAL_DEPLOYMENT`, che rappresenta la distribuzione locale sul dispositivo principale.

Questo problema può verificarsi nei seguenti casi:

- Il dispositivo principale è l'obiettivo di più distribuzioni con requisiti di versione dei componenti in conflitto. Ad esempio, il dispositivo principale potrebbe essere l'obiettivo di più distribuzioni che includono un `com.example.HelloWorld` componente, in cui una distribuzione richiede la versione 1.0.0 e l'altra richiede la versione 1.0.1. È impossibile avere un componente che soddisfi entrambi i requisiti, quindi l'implementazione non riesce.
- La versione del componente non esiste nel AWS IoT Greengrass servizio o nel dispositivo locale. Il componente potrebbe essere stato eliminato, ad esempio.
- Esistono versioni dei componenti che soddisfano i requisiti di versione, ma nessuna è compatibile con la piattaforma del dispositivo principale.
- La AWS IoT politica del dispositivo principale non concede l'`greengrass:ResolveComponentCandidates`autorizzazione. Cerca `Status Code: 403` nel registro degli errori per identificare il problema. Per risolvere il problema, aggiungi l'`greengrass:ResolveComponentCandidates`autorizzazione alla AWS IoT politica principale del dispositivo. Per ulteriori informazioni, consulta AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali.

Per risolvere questo problema, rivedi le distribuzioni per includere le versioni dei componenti compatibili o rimuovere quelle incompatibili. Per ulteriori informazioni su come rivedere le distribuzioni cloud, consulta. Rivedi le distribuzioni Per ulteriori informazioni su come rivedere le distribuzioni locali, consulta il comando AWS IoT Greengrass CLI deployment create.

software.amazon.awssdk.services.greengrassv2data.model.ResourceNotFoundE The latest version of Component <componentName> doesn't claim platform <coreDevicePlatform> compatibility

Potresti visualizzare questo errore quando distribuisci un componente su un dispositivo principale e il componente non elenca una piattaforma compatibile con la piattaforma del dispositivo principale. Esegui una di queste operazioni:

- Se il componente è un componente Greengrass personalizzato, puoi aggiornarlo per renderlo compatibile con il dispositivo principale. Aggiungi un nuovo manifesto che corrisponda alla piattaforma del dispositivo principale o aggiorna un manifesto esistente in modo che corrisponda alla piattaforma del dispositivo principale. Per ulteriori informazioni, consulta AWS IoT Greengrassriferimento alla ricetta del componente.
- Se il componente è fornito da AWS, controlla se un'altra versione del componente è compatibile con il dispositivo principale. Se nessuna versione è compatibile, contattaci AWS re:Postutilizzando il AWS IoT Greengrass tag o contattaci AWS Support.

com.aws.greengrass.componentmanager.exceptions.PackagingException: The deployment attempts to update the nucleus from aws.greengrass.Nucleus-<version> to aws.greengrass.Nucleus-<version> but no component of type nucleus was included as target component

Potresti visualizzare questo errore quando distribuisci un componente che dipende dal nucleo di Greengrass e il dispositivo principale esegue una versione di Greengrass nucleus precedente all'ultima versione secondaria disponibile. Questo errore si verifica perché il software AWS IoT Greengrass Core tenta di aggiornare automaticamente i componenti all'ultima versione compatibile. Tuttavia, il software AWS IoT Greengrass Core impedisce l'aggiornamento del nucleo Greengrass a una nuova versione secondaria, poiché diversi componenti AWS forniti dipendono da versioni minori specifiche del nucleo Greengrass. Per ulteriori informazioni, consulta Comportamento dell'aggiornamento del nucleo di Greengrass.

È necessario modificare la distribuzione per specificare la versione di Greengrass nucleus che si desidera utilizzare. Esegui una di queste operazioni:

- Rivedi la distribuzione per specificare la versione Greengrass nucleus attualmente in esecuzione sul dispositivo principale.

- Rivedi la distribuzione per specificare una versione secondaria successiva del nucleo Greengrass. Se si sceglie questa opzione, è necessario aggiornare anche le versioni di tutti i componenti AWS forniti che dipendono da specifiche versioni secondarie del nucleo Greengrass. Per ulteriori informazioni, consulta AWS-componenti forniti.

Error: com.aws.greengrass.deployment.exceptions.DeploymentException: Unable to process deployment. Greengrass launch directory is not set up or Greengrass is not set up as a system service

Potresti visualizzare questo errore quando sposti un dispositivo Greengrass da un gruppo di oggetti a un altro e poi di nuovo al gruppo originale con distribuzioni che richiedono il riavvio di Greengrass.

Per risolvere questo problema, ricrea la directory di avvio del dispositivo. Raccomandiamo inoltre vivamente l'aggiornamento alla versione 2.9.6 o successiva del nucleo Greengrass.

Quello che segue è uno script Linux per ricreare la directory di avvio. Salva lo script in un file chiamato`fix_directory.sh`.

```
#!/bin/bash

set -e

GG_ROOT=$1
GG_VERSION=$2

CURRENT="$GG_ROOT/alts/current"

if [ ! -L "$CURRENT" ]; then
  mkdir -p $GG_ROOT/alts/directory_fix
  echo "Relinking $GG_ROOT/alts/directory_fix to $CURRENT"
  ln -sf $GG_ROOT/alts/directory_fix $CURRENT
fi

TARGET=$(readlink $CURRENT)

if [[ ! -d "$TARGET" ]]; then
  echo "Creating directory: $TARGET"
  mkdir -p "$TARGET"
fi
```

```
DISTRO_LINK="$TARGET/distro"
DISTRO="$GG_ROOT/packages/artifacts-unarchived/aws.greengrass.Nucleus/$GG_VERSION/
aws.greengrass.nucleus/"
echo "Relinking Nucleus artifacts to $DISTRO_LINK"
ln -sf $DISTRO $DISTRO_LINK
```

Per eseguire lo script, esegui il seguente comando:

```
[root@ip-172-31-27-165 ~]# ./fix_directory.sh /greengrass/v2 2.9.5
Relinking /greengrass/v2/alts/directory_fix to /greengrass/v2/alts/current
Relinking Nucleus artifacts to /greengrass/v2/alts/directory_fix/distro
```

Info: com.aws.greengrass.deployment.exceptions.RetryableDeploymentDocumentDow Greengrass Cloud Service returned an error when getting full deployment configuration

È possibile che venga visualizzato questo errore quando il dispositivo principale riceve un documento di distribuzione di grandi dimensioni, ovvero un documento di distribuzione di dimensioni superiori a 7 KB (per distribuzioni destinate a oggetti) o 31 KB (per distribuzioni destinate a gruppi di oggetti). Per recuperare un documento di distribuzione di grandi dimensioni, la AWS IoT policy di un dispositivo principale deve consentire l'autorizzazione. `greengrass:GetDeploymentConfiguration` Questo errore può verificarsi quando il dispositivo principale non dispone di questa autorizzazione. Quando si verifica questo errore, la distribuzione riprova a tempo indeterminato e lo stato è In corso ()`IN_PROGRESS`.

Per risolvere questo problema, aggiungi l'`greengrass:GetDeploymentConfiguration`autorizzazione alla politica del AWS IoT dispositivo principale. Per ulteriori informazioni, consulta Aggiorna la politica di un dispositivo principale AWS IoT.

Warn: com.aws.greengrass.deployment.DeploymentService: Failed to get thing group hierarchy

Potresti visualizzare questo avviso quando il dispositivo principale riceve una distribuzione e la AWS IoT politica del dispositivo principale non consente l'`greengrass:ListThingGroupsForCoreDevice`autorizzazione. Quando si crea una distribuzione, il dispositivo principale utilizza questa autorizzazione per identificare i propri gruppi di

oggetti e rimuovere i componenti di qualsiasi gruppo di oggetti da cui è stato rimosso il dispositivo principale. Se il dispositivo principale esegue Greengrass nucleus v2.5.0, la distribuzione non riesce. Se il dispositivo principale esegue Greengrass nucleus v2.5.1 o versione successiva, la distribuzione procede ma non rimuove i componenti. Per ulteriori informazioni sul comportamento di rimozione dei gruppi di oggetti, vedere. Implementazione AWS IoT Greengrass dei componenti sui dispositivi

Per aggiornare il comportamento del dispositivo principale in modo da rimuovere i componenti per i gruppi di oggetti da cui rimuovi il dispositivo principale, aggiungi l'`greengrass:ListThingGroupsForCoreDevice`autorizzazione alla AWS IoT politica del dispositivo principale. Per ulteriori informazioni, consulta Aggiorna la politica di un dispositivo principale AWS IoT.

Info: com.aws.greengrass.deployment.DeploymentDocumentDownloader: Calling Greengrass cloud to get full deployment configuration

È possibile che questo messaggio informativo venga stampato più volte senza errori, poiché il dispositivo principale registra l'errore a livello di `DEBUG` registro. Questo problema può verificarsi quando il dispositivo principale riceve un documento di distribuzione di grandi dimensioni. Quando si verifica questo problema, la distribuzione riprova a tempo indeterminato e lo stato è In corso ()`IN_PROGRESS`. Per ulteriori informazioni su come risolvere questo problema, consulta questa voce sulla risoluzione dei problemi.

Caused by: software.amazon.awssdk.services.greengrassv2data.model.GreengrassV2DataE null (Service: GreengrassV2Data, Status Code: 403, Request ID: <some_request_id>, Extended Request ID: null)

Potresti visualizzare questo errore quando un'API dataplane non dispone `iot:Connect` dell'autorizzazione. Se non disponi della politica corretta, riceverai un. `GreengrassV2DataException: 403` Per creare una politica di autorizzazione, segui queste istruzioni:Creazione di una policy AWS IoT.

Problemi principali relativi ai componenti del dispositivo

Risolvi i problemi relativi ai componenti Greengrass sui dispositivi principali.

Argomenti

Warn: '<command>' is not recognized as an internal or external command

Potresti visualizzare questo errore nei log di un componente Greengrass quando il software AWS IoT Greengrass Core non riesce a eseguire un comando nello script del ciclo di vita del componente. Lo stato del componente diventa il risultato `BROKEN` di questo errore. Questo errore può verificarsi se l'utente di sistema che esegue il componente, ad esempio`ggc_user`, non riesce a trovare l'eseguibile del comando nelle cartelle del PATH.

Sui dispositivi Windows, verifica che la cartella che contiene l'eseguibile sia disponibile `PATH` per l'utente di sistema che esegue il componente. Se manca in`PATH`, esegui una delle seguenti operazioni:

- Aggiungi la cartella dell'eseguibile alla variabile di `PATH` sistema, che è disponibile per tutti gli utenti. Quindi, riavvia il componente.

Se si esegue Greengrass nucleus 2.5.0, dopo aver aggiornato la variabile di PATH sistema, è necessario riavviare il software AWS IoT Greengrass Core per eseguire i componenti con la versione aggiornata. PATH Se il software AWS IoT Greengrass Core non utilizza la versione aggiornata PATH dopo il riavvio del software, riavvia il dispositivo e riprova. Per ulteriori informazioni, consulta Esegui il software AWS IoT Greengrass Core.

- Aggiungi la cartella dell'eseguibile alla variabile PATH utente per l'utente di sistema che esegue il componente.

Lo script Python non registra i messaggi

I dispositivi core Greengrass raccolgono registri che puoi utilizzare per identificare problemi con i componenti. Se gli script `stdout` e i `stderr` messaggi Python non compaiono nei log dei componenti, potrebbe essere necessario svuotare il buffer o disabilitare il buffer per questi flussi di output standard in Python. Effettua una delle seguenti operazioni:

- Esegui Python con l'argomento -u per disabilitare il buffering su and. `stdout` `stderr`

  Linux or Unix

  ```
  python3 -u hello_world.py
  ```

  Windows

  ```
  py -3 -u hello_world.py
  ```

- Usa Setenv nella ricetta del tuo componente per impostare la variabile di ambiente PYTHONUNBUFFERED su una stringa non vuota. Questa variabile di ambiente disabilita il buffering su and. `stdout` `stderr`
- Svuota il buffer per i flussi or. `stdout` `stderr` Esegui una di queste operazioni:
  - Svuota un messaggio quando stampi.

    ```
    import sys

    print('Hello, error!', file=sys.stderr, flush=True)
    ```

  - Svuota un messaggio dopo la stampa. Puoi inviare più messaggi prima di scaricare lo stream.

    ```
    import sys
    ```

```
print('Hello, error!', file=sys.stderr)
sys.stderr.flush()
```

Per ulteriori informazioni su come verificare che lo script Python emetta messaggi di log, vedere. Monitora AWS IoT Greengrass i registri

La configurazione dei componenti non si aggiorna quando si modifica la configurazione predefinita

Quando si modifica la `DefaultConfiguration` ricetta di un componente, la nuova configurazione predefinita non sostituirà la configurazione esistente del componente durante una distribuzione. Per applicare la nuova configurazione predefinita, è necessario ripristinare la configurazione del componente alle impostazioni predefinite. Quando distribuisci il componente, specifica una singola stringa vuota come aggiornamento di ripristino.

Console

Reimposta i percorsi

```
[""]
```

AWS CLI

Il comando seguente crea una distribuzione su un dispositivo principale.

```
aws greengrassv2 create-deployment --cli-input-json file://reset-configuration-
deployment.json
```

Il `reset-configuration-deployment.json` file contiene il seguente documento JSON.

```
{
  "targetArn": "arn:aws:iot:us-west-2:123456789012:thing/MyGreengrassCore",
  "deploymentName": "Deployment for MyGreengrassCore",
  "components": {
    "com.example.HelloWorld": {
      "componentVersion": "1.0.0",
      "configurationUpdate": {,
        "reset": [""]
```

```
      }
    }
  }
}
```

Greengrass CLI

Il seguente comando Greengrass CLI crea una distribuzione locale su un dispositivo principale.

```
sudo greengrass-cli deployment create \
  --recipeDir recipes \
  --artifactDir artifacts \
  --merge "com.example.HelloWorld=1.0.0" \
  --update-config reset-configuration-deployment.json
```

Il `reset-configuration-deployment.json` file contiene il seguente documento JSON.

```
{
  "com.example.HelloWorld": {
    "RESET": [""]
  }
}
```

awsiot.greengrasscoreipc.model.UnauthorizedError

Potresti visualizzare questo errore nei log di un componente Greengrass quando il componente non dispone dell'autorizzazione per eseguire un'operazione IPC su una risorsa. Per concedere a un componente l'autorizzazione a chiamare un'operazione IPC, definite una politica di autorizzazione IPC nella configurazione del componente. Per ulteriori informazioni, consulta Autorizza i componenti a eseguire operazioni IPC.

> **Tip**
>
> Se modificate la `DefaultConfiguration` ricetta di un componente, dovete ripristinare la configurazione del componente alla nuova configurazione predefinita. Quando distribuite il componente, specificate una singola stringa vuota come aggiornamento di ripristino. Per ulteriori informazioni, consulta La configurazione dei componenti non si aggiorna quando si modifica la configurazione predefinita.

com.aws.greengrass.authorization.exceptions.AuthorizationException: Duplicate policy ID "<id>" for principal "<componentList>"

Potresti visualizzare questo errore se più politiche di autorizzazione IPC, inclusi tutti i componenti del dispositivo principale, utilizzano lo stesso ID di policy.

Controlla le politiche di autorizzazione IPC dei tuoi componenti, correggi eventuali duplicati e riprova. Per creare ID di policy univoci, ti consigliamo di combinare il nome del componente, il nome del servizio IPC e un contatore. Per ulteriori informazioni, consulta Autorizza i componenti a eseguire operazioni IPC.

> **Tip**
>
> Se si modifica la `DefaultConfiguration` ricetta di un componente, è necessario ripristinare la configurazione del componente alla nuova configurazione predefinita. Quando distribuite il componente, specificate una singola stringa vuota come aggiornamento di ripristino. Per ulteriori informazioni, consulta La configurazione dei componenti non si aggiorna quando si modifica la configurazione predefinita.

com.aws.greengrass.tes.CredentialRequestHandler: Error in retrieving AwsCredentials from TES (HTTP 400)

Potresti visualizzare questo errore quando un dispositivo principale non riesce a ottenere AWS le credenziali dal servizio di scambio di token. Il codice di stato HTTP 400 indica che questo errore si è verificato perché il ruolo IAM per lo scambio di token del dispositivo principale non esiste o non esiste una relazione di fiducia che consenta al provider di AWS IoT credenziali di assumerlo.

Esegui questa operazione:

1. Identifica il ruolo di scambio di token utilizzato dal dispositivo principale. Il messaggio di errore include l'alias del AWS IoT ruolo del dispositivo principale, che rimanda al ruolo di scambio di token. Esegui il comando seguente sul tuo computer di sviluppo e sostituiscilo *`MyGreengrassCoreTokenExchangeRoleAlias`*con il nome dell'alias di AWS IoT ruolo contenuto nel messaggio di errore.

   ```
   aws iot describe-role-alias --role-alias MyGreengrassCoreTokenExchangeRoleAlias
   ```

La risposta include l'Amazon Resource Name (ARN) del ruolo IAM di token exchange.

```
{
  "roleAliasDescription": {
    "roleAlias": "MyGreengrassCoreTokenExchangeRoleAlias",
    "roleAliasArn": "arn:aws:iot:us-west-2:123456789012:rolealias/
MyGreengrassCoreTokenExchangeRoleAlias",
    "roleArn": "arn:aws:iam::123456789012:role/MyGreengrassV2TokenExchangeRole",
    "owner": "123456789012",
    "credentialDurationSeconds": 3600,
    "creationDate": "2021-02-05T16:46:18.042000-08:00",
    "lastModifiedDate": "2021-02-05T16:46:18.042000-08:00"
  }
}
```

2. Verifica che il ruolo esista. Esegui il comando seguente e sostituisci *MyGreengrassV2TokenExchangeRole* con il nome del ruolo di scambio di token.

```
aws iam get-role --role-name MyGreengrassV2TokenExchangeRole
```

Se il comando restituisce un `NoSuchEntity` errore, il ruolo non esiste e devi crearlo. Per ulteriori informazioni su come creare e configurare questo ruolo, vedereAutorizza i dispositivi principali a interagire conAWSservizi.

3. Verifica che il ruolo abbia una relazione di fiducia che consenta al fornitore delle AWS IoT credenziali di assumerlo. La risposta del passaggio precedente contiene un`AssumeRolePolicyDocument`, che definisce le relazioni di fiducia del ruolo. Il ruolo deve definire una relazione di fiducia che `credentials.iot.amazonaws.com` consenta di assumerlo. Questo documento dovrebbe essere simile all'esempio seguente.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "credentials.iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

Se le relazioni di fiducia del ruolo non `credentials.iot.amazonaws.com` consentono di assumerlo, è necessario aggiungere questa relazione di fiducia al ruolo. Per ulteriori informazioni, consulta Modifica di un ruolo nella AWS Identity and Access Management Guida per l'utente di IAM.

com.aws.greengrass.tes.CredentialRequestHandler: Error in retrieving AwsCredentials from TES (HTTP 403)

Potresti visualizzare questo errore quando un dispositivo principale non riesce a ottenere AWS le credenziali dal servizio di scambio di token. Il codice di stato HTTP 403 indica che questo errore si è verificato perché le AWS IoT politiche del dispositivo principale non concedono l'`iot:AssumeRoleWithCertificate`autorizzazione per l'alias di AWS IoT ruolo del dispositivo principale.

Esamina AWS IoT le politiche del dispositivo principale e aggiungi l'`iot:AssumeRoleWithCertificate`autorizzazione per l'alias di AWS IoT ruolo del dispositivo principale. Il messaggio di errore include l'alias del AWS IoT ruolo corrente del dispositivo principale. Per ulteriori informazioni su questa autorizzazione e su come aggiornare le AWS IoT politiche del dispositivo principale, consulta AWS IoTPolitica minima per i dispositivi AWS IoT Greengrass V2 principali eAggiorna la politica di un dispositivo principale AWS IoT.

com.aws.greengrass.tes.CredentialsProviderError: Could not load credentials from any providers

Potresti visualizzare questo errore quando il componente tenta di richiedere le AWS credenziali e non riesce a connettersi al servizio di scambio di token.

Esegui questa operazione:

- Verifica che il componente dichiari una dipendenza dal componente del servizio di scambio di token,. `aws.greengrass.TokenExchangeService` In caso contrario, aggiungi la dipendenza e ridistribuisci il componente.
- Se il componente viene eseguito in docker, assicurati di applicare le impostazioni di rete e le variabili di ambiente corrette, in base a. Usa AWS le credenziali nei componenti del contenitore Docker (Linux)

- Se il componente è scritto in NodeJS, imposta dns. setDefaultResultOrdina a. **`ipv4first`**
- `/etc/hosts`Cerca una voce che inizi con `::1` e contenga`localhost`. Rimuovi la voce per vedere se ha causato la connessione del componente al servizio di scambio di token all'indirizzo sbagliato.

Received error when attempting to retrieve ECS metadata: Could not connect to the endpoint URL: "<tokenExchangeServiceEndpoint>"

Potresti visualizzare questo errore quando il componente non esegue il servizio di scambio di token e un componente tenta di richiedere le AWS credenziali.

Esegui questa operazione:

- Verifica che il componente dichiari una dipendenza dal componente del servizio di scambio di token,. `aws.greengrass.TokenExchangeService` In caso contrario, aggiungi la dipendenza e ridistribuisci il componente.
- Verifica se il componente utilizza le AWS credenziali nel suo ciclo di vita. `install` AWS IoT Greengrass non garantisce la disponibilità del servizio di scambio di token durante il ciclo di vita. `install` Aggiorna il componente per spostare il codice che utilizza AWS le credenziali nel `run` ciclo di vita `startup` o, quindi ridistribuisci il componente.

copyFrom: <configurationPath> is already a container, not a leaf

Potresti visualizzare questo errore quando modifichi un valore di configurazione da un tipo di contenitore (un elenco o un oggetto) a un tipo non contenitore (una stringa, un numero o un valore booleano). Esegui questa operazione:

1. Controlla la ricetta del componente per vedere se la sua configurazione predefinita imposta quel valore di configurazione su un elenco o un oggetto. In tal caso, rimuovi o modifica quel valore di configurazione.
2. Crea una distribuzione per ripristinare il valore di configurazione al valore predefinito. Per ulteriori informazioni, consulta Creare distribuzione e Aggiornamento delle configurazioni dei componenti.

Quindi, puoi impostare quel valore di configurazione su una stringa, un numero o un valore booleano.

com.aws.greengrass.componentmanager.plugins.docker.exceptions.DockerLoginException: Error logging into the registry using credentials - 'The stub received bad data.'

Potresti visualizzare questo errore nei registri del nucleo di Greengrass quando il componente Docker Application Manager tenta di scaricare un'immagine Docker da un repository privato in Amazon Elastic Container Registry (Amazon ECR). Questo errore si verifica se si utilizza l'helper per le credenziali Docker (). `wincred` `docker-credential-wincred` Di conseguenza, Amazon ECR non è in grado di memorizzare le credenziali di accesso.

Esegui una delle seguenti azioni:

- Se non usi l'helper per le credenziali `wincred` Docker, rimuovi il `docker-credential-wincred` programma dal dispositivo principale.
- Se usi l'helper per le credenziali `wincred` Docker, procedi come segue:
  1. Rinomina il `docker-credential-wincred` programma sul dispositivo principale. `wincred`Sostituiscilo con un nuovo nome per l'helper delle credenziali di Windows Docker. Ad esempio, puoi rinominarlo in. `docker-credential-wincredreal`
  2. Aggiorna l'`credsStore`opzione nel file di configurazione Docker (`.docker/config.json`) per utilizzare il nuovo nome per l'helper delle credenziali di Windows Docker. Ad esempio, se hai rinominato il programma in`docker-credential-wincredreal`, aggiorna l'opzione in. `credsStore` `wincredreal`

     ```
     {
       "credsStore": "wincredreal"
     }
     ```

java.io.IOException: Cannot run program "cmd" ...: [LogonUser] The password for this account has expired.

Potresti visualizzare questo errore su un dispositivo Windows Core quando l'utente di sistema che esegue i processi del componente, ad esempio`ggc_user`, ha una password scaduta. Di conseguenza, il software AWS IoT Greengrass Core non è in grado di eseguire i processi dei componenti come utente del sistema.

Per aggiornare la password di un utente del sistema Greengrass

1. Esegui il seguente comando come amministratore per impostare la password dell'utente. Sostituisci *ggc_user con l'utente* di sistema e sostituisci *la password con la password* da impostare.

   ```
   net user ggc_user password
   ```

2. Utilizzate l'PsExec utilità per memorizzare la nuova password dell'utente nell'istanza di Credential Manager per l'account. LocalSystem Sostituisci *la password* con la password dell'utente che hai impostato.

   ```
   psexec -s cmd /c cmdkey /generic:ggc_user /user:ggc_user /pass:password
   ```

> **Tip**
>
> A seconda della configurazione di Windows, la password dell'utente potrebbe essere impostata per scadere in date future. Per garantire che le tue applicazioni Greengrass continuino a funzionare, tieni traccia della scadenza della password e aggiornala prima che scada. Puoi anche impostare la password dell'utente in modo che non scada mai.
>
> - Per verificare la scadenza di un utente e della relativa password, esegui il comando seguente.
>
>   ```
>   net user ggc_user | findstr /C:expires
>   ```
>
> - Per impostare la password di un utente in modo che non scada mai, esegui il comando seguente.
>
>   ```
>   wmic UserAccount where "Name='ggc_user'" set PasswordExpires=False
>   ```
>
> - Se utilizzi Windows 10 o versioni successive in cui il wmiccomando è obsoleto, esegui il comando seguente. PowerShell
>
>   ```
>   Get-CimInstance -Query "SELECT * from Win32_UserAccount WHERE name =
>    'ggc_user'" | Set-CimInstance -Property @{PasswordExpires="False"}
>   ```

aws.greengrass.StreamManager: Instant exceeds minimum or maximum instant

Quando aggiorni stream manager v2.0.7 a una versione compresa tra v2.0.8 e v2.0.11, potresti visualizzare il seguente errore nei log del componente stream manager se il componente non si avvia.

```
2021-07-16T00:54:58.568Z [INFO] (Copier) aws.greengrass.StreamManager:
 stdout. Caused by: com.fasterxml.jackson.databind.JsonMappingException:
 Instant exceeds minimum or maximum instant (through reference chain:
 com.amazonaws.iot.greengrass.streammanager.export.PersistedSuccessExportStatesV1["lastExportTi
 {scriptName=services.aws.greengrass.StreamManager.lifecycle.startup.script,
 serviceName=aws.greengrass.StreamManager, currentState=STARTING}
2021-07-16T00:54:58.579Z [INFO] (Copier) aws.greengrass.StreamManager: stdout.
 Caused by: java.time.DateTimeException: Instant exceeds minimum or maximum instant.
 {scriptName=services.aws.greengrass.StreamManager.lifecycle.startup.script,
 serviceName=aws.greengrass.StreamManager, currentState=STARTING}
```

Se hai distribuito stream manager v2.0.7 e desideri eseguire l'aggiornamento a una versione successiva, devi eseguire l'aggiornamento direttamente a stream manager v2.0.12. Per ulteriori informazioni sul componente stream manager, consulta. Stream manager

Problemi relativi ai componenti della funzione Lambda del dispositivo principale

Risolvi i problemi relativi ai componenti della funzione Lambda sui dispositivi principali.

Argomenti

- The following cgroup subsystems are not mounted: devices, memory
- ipc_client.py:64,HTTP Error 400:Bad Request, b'No subscription exists for the source <label-or-lambda-arn> and subject <label-or-lambda-arn>

The following cgroup subsystems are not mounted: devices, memory

Potresti visualizzare questo errore quando esegui una funzione Lambda containerizzata nei seguenti casi:

- Il dispositivo principale non ha cgroup v1 abilitato per la memoria o i cgroup del dispositivo.

- Il dispositivo principale ha cgroups v2 abilitato. Le funzioni Greengrass Lambda richiedono cgroups v1 e cgroups v1 e v2 si escludono a vicenda.

Per abilitare cgroups v1, avvia il dispositivo con i seguenti parametri del kernel Linux.

```
cgroup_enable=memory cgroup_memory=1 systemd.unified_cgroup_hierarchy=0
```

> **Tip**
>
> Su un Raspberry Pi, modifica il `/boot/cmdline.txt` file per impostare i parametri del kernel del dispositivo.

ipc_client.py:64,HTTP Error 400:Bad Request, b'No subscription exists for the source <label-or-lambda-arn> and subject <label-or-lambda-arn>

Potresti visualizzare questo errore quando esegui una funzione Lambda V1, che utilizza AWS IoT Greengrass Core SDK, su un dispositivo core V2 senza specificare un abbonamento nel componente legacy del router di abbonamento. Per risolvere questo problema, distribuisci e configura il router di abbonamento legacy per specificare gli abbonamenti richiesti. Per ulteriori informazioni, consulta Importa funzioni Lambda V1.

La versione del componente è stata interrotta

Potresti visualizzare una notifica sulla tua Personal Health Dashboard (PHD) quando una versione di un componente sul tuo dispositivo principale viene interrotta. La versione del componente invia questa notifica al tuo dottorato di ricerca entro 60 minuti dalla cessazione della produzione.

Per vedere quali implementazioni devi rivedere, procedi come segue utilizzando: AWS Command Line Interface

1. Esegui il comando seguente per ottenere un elenco dei tuoi dispositivi principali.

   ```
   aws greengrassv2 list-core-devices
   ```

2. Esegui il comando seguente per recuperare lo stato dei componenti su ciascun dispositivo principale dal passaggio 1. Sostituiscilo *coreDeviceName* con il nome di ogni dispositivo principale da interrogare.

```
aws greengrassv2 list-installed-components --core-device-thing-name coreDeviceName
```

3. Raccogli i dispositivi principali con la versione del componente fuori produzione installata nei passaggi precedenti.
4. Esegui il comando seguente per recuperare lo stato di tutti i processi di distribuzione per ogni dispositivo principale dal Passaggio 3. Sostituiscilo *coreDeviceName* con il nome del dispositivo principale da interrogare.

   ```
   aws greengrassv2 list-effective-deployments --core-device-thing-name coreDeviceName
   ```

   La risposta contiene l'elenco dei processi di distribuzione per il dispositivo principale. È possibile modificare la distribuzione per scegliere un'altra versione del componente. Per ulteriori informazioni su come rivedere una distribuzione, consulta Revisionare le distribuzioni.

Problemi relativi all'interfaccia a riga di comando di Greengrass

Risolvi i problemi con la CLI di Greengrass.

Argomenti

- java.lang.RuntimeException: Unable to create ipc client

java.lang.RuntimeException: Unable to create ipc client

Potresti visualizzare questo errore quando esegui un comando Greengrass CLI e specifichi una cartella principale diversa da quella in cui è installato il software AWS IoT Greengrass Core.

Effettuate una delle seguenti operazioni per impostare il percorso principale e sostituitelo */greengrass/v2* con il percorso di installazione del software AWS IoT Greengrass Core:

- Impostare la variabile di ambiente `GGC_ROOT_PATH` su */greengrass/v2*.
- Aggiungete l'`--ggcRootPath` */greengrass/v2*argomento al comando come illustrato nell'esempio seguente.

  ```
  greengrass-cli --ggcRootPath /greengrass/v2 <command> <subcommand> [arguments]
  ```

AWS Command Line Interface problemi

Risolvi i AWS CLI problemi relativi a. AWS IoT Greengrass V2

Argomenti

Error: Invalid choice: 'greengrassv2'

È possibile che venga visualizzato questo errore quando si esegue un AWS IoT Greengrass V2 comando con AWS CLI (ad esempio,`aws greengrassv2 list-core-devices`).

Questo errore indica che hai una versione di AWS CLI che non supporta AWS IoT Greengrass V2. Per utilizzarlo AWS IoT Greengrass V2 con AWS CLI, è necessario disporre di una delle seguenti versioni o successive:

- Versione minima AWS CLI V1: v1.18.197
- Versione minima AWS CLI V2: v2.1.11

> **Tip**
>
> Puoi eseguire il seguente comando per verificare la versione di cui disponi AWS CLI .
>
> ```
> aws --version
> ```

Per risolvere questo problema, aggiornalo AWS CLI a una versione successiva che supporti AWS IoT Greengrass V2. Per ulteriori informazioni, vedere Installazione, aggiornamento e disinstallazione di AWS CLI nella Guida per l'AWS Command Line Interface utente.

Codici di errore di distribuzione dettagliati

Usa i codici di errore e le soluzioni in queste sezioni per risolvere i problemi di distribuzione dei componenti quando usi Greengrass nucleus versione 2.8.0 o successiva.

Il nucleo di Greengrass riporta gli errori di distribuzione come una gerarchia dal codice meno specifico al più specifico disponibile. È possibile utilizzare questa gerarchia per individuare il motivo di un errore di distribuzione. Ad esempio, la seguente è una possibile gerarchia degli errori:

- FALLIMENTO_DI INSTALLAZIONE
  - ARTIFACT_DOWNLOAD_ERROR
    - IO_ERROR
      - SPAZIO SU DISCO CRITICO

I codici di errore sono organizzati in tipi. Ogni tipo rappresenta una classe di errori che possono verificarsi.AWS IoT Greengrasssegnala questi tipi di errori nella console, nell'API eAWS CLI. Può essere presente più di un tipo di errore, a seconda degli errori segnalati nella gerarchia degli errori. Nell'esempio precedente, il tipo di errore restituito è`DEVICE_ERROR`.

Le tipologie sono:

- ERRORE_DI AUTORIZZAZIONE— L'accesso a un'operazione che richiede l'autorizzazione è stato negato.
- RICHIESTA_ERRORE— Si è verificato un errore a causa di un problema nel documento di distribuzione.
- ERRORE DELLA RICETTA DEL COMPONENTE— Si è verificato un errore a causa di un problema nella composizione di un componente.
- AWS_COMPONENT_ERROR— Si è verificato un errore durante l'avvio o la rimozione di unAWScomponente fornito.
- ERRORE_COMPONENT_UTENTE— Si è verificato un errore durante l'avvio o la rimozione di un componente utente.
- ERRORE DEL COMPONENTE— Si è verificato un errore durante l'avvio o la rimozione di un componente, ma il nucleo di Greengrass non è stato in grado di determinare se il componente è unAWScomponente fornito o componente utente.
- ERRORE_DISPOSITIVO— Si è verificato un errore con l'I/O locale o si è verificato un altro errore del dispositivo.
- ERRORE_DI DIPENDENZA— Una distribuzione non è riuscita a scaricare un artefatto da Amazon S3 o a recuperare un'immagine da un registro ECR.
- HTTP_ERROR— Si è verificato un errore con una richiesta HTTP.
- ERRORE_DI RETE— Si è verificato un errore con la rete del dispositivo.

- ERRORE DEL NUCLEO— Il nucleo di Greengrass non è riuscito a localizzare un componente o non è riuscito a trovare la versione del nucleo attivo.
- ERRORE_SERVER— Un server ha restituito un errore 500 in risposta a una richiesta.
- ERRORE DEL SERVIZIO CLOUD— Si è verificato un errore conAWS IoT Greengrassservizio cloud.
- ERRORE_SCONOSCIUTO— Un'eccezione deselezionata è stata generata dal componente.

Molti degli errori in questa sezione riportano informazioni aggiuntive nelAWS IoT GreengrassRegistri principali. Questi registri vengono archiviati nel file system locale del dispositivo principale. Ci sono registri perAWS IoT GreengrassSoftware di base e per ogni singolo componente. Per informazioni sull'accesso ai registri, vedereAccedere ai log del file system.

Errore di autorizzazione

ACCESS_NEGATO

Potresti ricevere questo errore quandoAWSl'operazione del servizio restituisce un errore 403 perché le autorizzazioni non sono impostate correttamente. Controlla il codice di errore più specifico per i dettagli.

GET_DEPLOYMENT_CONFIGURATION_ACCESS_DENIED

Potresti ricevere questo errore quandoAWS IoTla politica non consente il permesso di chiamare il`GetDeploymentConfiguration`operazione. Aggiungi la`greengrass::GetDeploymentConfiguration`autorizzazione alla politica del dispositivo principale.

GET_COMPONENT_VERSION_ARTIFACT_ACCESS_DENIED

Potresti ricevere questo errore quando il dispositivo principaleAWS IoTla politica non consente il`greengrass:GetComponentVersionArtifact`permesso. Aggiungi l'autorizzazione alla politica del dispositivo principale.

RESOLVE_COMPONENT_CANDIDATI_ACCESS_DENIED

Potresti ricevere questo errore quando il dispositivo principaleAWS IoTla politica non consente il`greengrass:ResolveComponentCandidates`permesso. Aggiungi l'autorizzazione alla politica del dispositivo principale.

GET_ECR_CREDENTIAL_ERROR

Potresti ricevere questo errore quando la distribuzione non è riuscita ad autenticarsi con un registro privato in ECR. Controlla nel registro la presenza di un errore specifico, quindi riprova a eseguire la distribuzione.

UTENTE_NON AUTORIZZATO PER_DOCKER

Potresti ricevere questo errore quando l'utente Greengrass non è autorizzato a utilizzare Docker. Assicurati di eseguire Greengrass come utente root o che l'utente sia aggiunto al`docker`gruppo. Quindi riprova a eseguire la distribuzione.

S3_ACCESS_NEGATO

Potresti ricevere questo errore quando un'operazione Amazon S3 restituisce un errore 403. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

S3_HEAD_OBJECT_ACCESS NEGATO

Potresti ricevere questo errore anche quando il ruolo di scambio di token del dispositivo non consenteAWS IoT GreengrassSoftware di base per scaricare l'artefatto del componente dall'URL dell'oggetto S3 specificato nella ricetta del componente o che l'artefatto del componente non è disponibile. Verifica che il ruolo di scambio di token lo consenta`s3:GetObject`per l'URL dell'oggetto S3 in cui l'artefatto è disponibile e in cui l'artefatto è presente.

S3_GET_BUCKET_LOCATION_ACCESS_DENIED

Potresti ricevere questo errore quando il ruolo di scambio di token del dispositivo non consente`s3:GetBucketLocation`autorizzazione per il bucket Amazon S3 in cui l'artefatto è disponibile. Verifica che il dispositivo consenta l'autorizzazione, quindi riprova a eseguire l'installazione.

S3_GET_OBJECT_ACCESS_DENIED

Potresti ricevere questo errore anche quando il ruolo di scambio di token del dispositivo non consenteAWS IoT GreengrassSoftware di base per scaricare l'artefatto del componente dall'URL dell'oggetto S3 specificato nella ricetta del componente o che l'artefatto del componente non è disponibile. Verifica che il ruolo di scambio di token lo consenta`s3:GetObject`per l'URL dell'oggetto S3 in cui l'artefatto è disponibile e in cui l'artefatto è presente.

Errore nella richiesta

NUCLEUS_CAPACITÀ_RICHIESTI_MANCANTI

Potresti ricevere questo errore quando la versione nucleus della distribuzione non è in grado di eseguire un'operazione richiesta, come il download di una configurazione di grandi dimensioni o l'impostazione dei limiti delle risorse Linux. Riprova la distribuzione con una versione di Nucleus che supporti l'operazione.

ERRORE_NUCLEUS_RISOLTO

È possibile che venga visualizzato questo errore quando una distribuzione tenta di distribuire più componenti nucleus. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

ERRORE_CIRCOLARE_DIPENDENZA_COMPONENTE

È possibile che venga visualizzato questo errore quando due componenti della distribuzione dipendono l'uno dall'altro. Rivedi la configurazione dei componenti in modo che i componenti della distribuzione non dipendano l'uno dall'altro.

AGGIORNAMENTO NON AUTORIZZATO DELLA VERSIONE DI NUCLEUS_MINOR_NUCLEUS_MINOR_

Potresti ricevere questo errore quando un componente della tua distribuzione richiede un aggiornamento della versione secondaria di Nucleus, ma tale versione non è specificata nella distribuzione. Ciò consente di ridurre gli aggiornamenti accidentali delle versioni secondarie per i componenti che dipendono da una versione diversa. Includi la nuova versione di Minor Nucleus nella distribuzione.

MANAGER_DOCKER_APPLICATION_MANCANTE

Potresti ricevere questo errore quando distribuisci un componente Docker senza distribuire il gestore applicazioni Docker. Assicurati che la tua distribuzione includa il gestore delle applicazioni Docker.

SERVIZIO_TOKEN_EXCHANGE_MANCANTE

Potresti ricevere questo errore quando la distribuzione desidera scaricare un artefatto di immagine Docker da un registro ECR privato senza implementare il servizio di scambio di token. Assicurati che la distribuzione includa il servizio di scambio di token.

REQUISITI_DELLA VERSIONE DEI COMPONENTI NON SODDISFATTI

È possibile che venga visualizzato questo errore quando c'è un conflitto di vincoli di versione o se la versione di un componente non esiste. Per ulteriori informazioni, consulta Error: com.aws.greengrass.componentmanager.exceptions.NoAvailableComponentVersionException: Failed to negotiate component <name> version with cloud and no local applicable version satisfying requirement <requirements>.

THROTTLING_ERROR

Potresti ricevere questo errore quandoAWSIl funzionamento del servizio ha superato una quota tariffaria. Riprova a eseguire la distribuzione.

RICHIESTA_CONFLITTO

Potresti ricevere questo errore quandoAWSl'operazione del servizio restituisce un errore 409 perché la distribuzione sta tentando di eseguire più di un'operazione alla volta. Riprova a eseguire la distribuzione.

RISORSA_NON TROVATA

Potresti ricevere questo errore quandoAWSl'operazione del servizio restituisce un errore 404 perché non è stata trovata una risorsa. Controlla nel registro la risorsa mancante.

ESEGUI CON CONFIG_NOT_VALID

Potresti ricevere questo errore quando`posixUser`,`posixGroup`, oppure`windowsUser`le informazioni specificate per eseguire il componente non sono valide. Verifica che l'utente sia valido, quindi riprova la distribuzione.

REGIONE_NON SUPPORTATA

Potresti ricevere questo errore quando la regione specificata per la distribuzione non è supportata daAWS IoT Greengrass. Controlla la regione e riprova a eseguire la distribuzione.

IOT_CRED_ENDPOINT_NON VALIDO

Potresti ricevere questo errore quandoAWS IoTl'endpoint credenziale specificato nella configurazione non è valido. Controlla l'endpoint e riprova la richiesta.

IOT_DATA_ENDPOINT_NON VALIDO

Potresti ricevere questo errore quandoAWS IoTl'endpoint dei dati specificato nella configurazione non è valido. Controlla l'endpoint e riprova la richiesta.

S3_HEAD_OBJECT_RESOURCE_NON TROVATA

Potresti ricevere questo errore quando l'artefatto del componente non è disponibile nell'URL dell'oggetto S3 specificato nella ricetta del componente. Verifica di aver caricato l'artefatto nel bucket S3 e che l'URI dell'artefatto corrisponda all'URL dell'oggetto S3 dell'artefatto nel bucket.

S3_GET_BUCKET_LOCATION_RESOURCE_NON TROVATA

Potresti ricevere questo errore quando il bucket Amazon S3 non viene trovato. Verifica che il bucket esista e riprova la distribuzione.

S3_GET_OBJECT_RESOURCE_NON TROVATA

Potresti ricevere questo errore quando l'artefatto del componente non è disponibile nell'URL dell'oggetto S3 specificato nella ricetta del componente. Verifica di aver caricato l'artefatto nel bucket S3 e che l'URI dell'artefatto corrisponda all'URL dell'oggetto S3 dell'artefatto nel bucket.

IO_MAPPING_ERROR

È possibile che questo errore venga visualizzato quando si verifica un errore di I/O durante l'analisi del documento o della ricetta di distribuzione. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

Errore nella ricetta del componente

ERRORE DI ANALISI DELLA RICETTA

Potresti ricevere questo errore quando la ricetta di distribuzione non può essere analizzata a causa di un errore nella struttura della ricetta. Verifica che la ricetta sia formattata correttamente e riprova a eseguire la distribuzione.

ERRORE DI ANALISI DEI METADATI DELLA RICETTA

Potresti ricevere questo errore quando i metadati della ricetta di distribuzione scaricati dal cloud non possono essere analizzati. Contattare AWS Support.

ARTEFATTO_URI_NON VALIDO

Potresti ricevere questo errore quando l'URI di un artefatto in una ricetta non è formattato correttamente. Controlla nel registro l'URI non valido, aggiorna l'URI nella ricetta, quindi riprova a eseguire la distribuzione.

S3_ARTIFACT_URI_NON VALIDO

Potresti ricevere questo errore quando l'URI Amazon S3 di un artefatto in una ricetta non è valido. Controlla nel registro l'URI non valido, aggiorna l'URI nella ricetta, quindi riprova a eseguire la distribuzione.

DOCKER_ARTIFACT_URI_NOT_VALID

Potresti ricevere questo errore quando l'URI Docker di un artefatto in una ricetta non è valido. Controlla nel registro l'URI non valido, aggiorna l'URI nella ricetta, quindi riprova a eseguire la distribuzione.

ARI_ARTEFATTO VUOTO

Potresti ricevere questo errore quando l'URI di un artefatto non è specificato in una ricetta. Controlla nel registro l'elemento a cui manca un URI, aggiorna l'URI nella ricetta, quindi riprova la distribuzione.

SCHEMA DI ARTEFATTO VUOTO

Potresti ricevere questo errore quando uno schema URI non è definito per un artefatto. Controlla nel registro l'URI non valido, aggiorna l'URI nella ricetta, quindi riprova a eseguire la distribuzione.

SCHEMA_ARTE_DI_ARTEFATTO NON SUPPORTATO

Potresti ricevere questo errore quando uno schema URI non è supportato dalla versione di Nucleus in esecuzione. Un URI non è valido o è necessario aggiornare la versione di Nucleus. Se l'URI non è valido, controlla nel registro l'URI non valido, aggiorna l'URI nella ricetta, quindi riprova la distribuzione.

MANIFESTO DELLA RICETTA MANCANTE

Potresti ricevere questo errore quando la sezione manifest non è inclusa nella ricetta. Aggiungi il manifest alla ricetta e riprova la distribuzione.

ALGORITMO_ARTEFATTO MANCANTE DELLA RICETTA

Potresti ricevere questo errore quando un artefatto non locale viene specificato all'interno di una ricetta senza un algoritmo di hash. Aggiungete l'algoritmo all'artefatto e riprovate con la richiesta.

ARTEFATTO_CHECKSUM_MISMATCH

Potresti ricevere questo errore quando un artefatto scaricato ha un digest diverso da quello specificato nella ricetta. Assicurati che la ricetta contenga il riassunto

corretto, quindi riprova a implementare. Per ulteriori informazioni, consulta Error: com.aws.greengrass.componentmanager.exceptions.ArtifactChecksumMismatchException: Integrity check for downloaded artifact failed. Probably due to file corruption..

DIPENDENZA_COMPONENTE NON VALIDA

È possibile che venga visualizzato questo errore quando il tipo di dipendenza specificato in una ricetta di distribuzione non è valido. Controlla la ricetta e poi riprova con la tua richiesta.

CONFIG_INTERPOLATE_ERROR

Potresti ricevere questo errore durante l'interpolazione di una variabile di ricetta. Controlla il registro per i dettagli.

IO_MAPPING_ERROR

È possibile che questo errore venga visualizzato quando si verifica un errore di I/O durante l'analisi del documento o della ricetta di distribuzione. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

AWSerrore del componente, errore del componente utente, errore del componente

I seguenti codici di errore vengono restituiti quando si verifica un problema con un componente. Il tipo di errore effettivo segnalato dipende dal componente specifico che ha generato l'errore. Se il nucleo di Greengrass identifica il componente come fornito daAWS IoT Greengrass, restituisce`AWS_COMPONENT_ERROR`. Se il componente viene identificato come componente utente, il nucleo di Greengrass ritorna`USER_COMPONENT_ERROR`. Se il nucleo di Greengrass non può dirlo, ritorna`COMPONENT_ERROR`.

ERRORE DI AGGIORNAMENTO DEI COMPONENTI

Potresti ricevere questo errore quando un componente non viene aggiornato durante una distribuzione. Controlla eventuali codici di errore aggiuntivi o controlla il registro per vedere cosa ha causato l'errore.

COMPONENTE ROTTO

È possibile che venga visualizzato questo errore quando un componente si rompe durante una distribuzione. Controllate il registro dei componenti per i dettagli degli errori, quindi riprovate la distribuzione.

RIMUOVI L'ERRORE DEL COMPONENTE

Potresti ricevere questo errore quando il nucleo non è in grado di rimuovere un componente durante una distribuzione. Controlla il registro per i dettagli degli errori, quindi riprova a eseguire la distribuzione.

TIMEOUT DEL COMPONENTE BOOTSTRAP_OUT

Potresti ricevere questo errore quando l'attività di bootstrap di un componente ha richiesto più tempo del timeout configurato. Aumenta il timeout o riduci il tempo di esecuzione dell'attività di bootstrap, quindi riprova a eseguire la distribuzione.

ERRORE_BOOTSTRAP_COMPONENTE

Potresti ricevere questo errore quando l'attività di bootstrap di un componente presenta un errore. Controlla il registro per i dettagli degli errori, quindi riprova a eseguire la distribuzione.

CONFIGURAZIONE_COMPONENTE NON VALIDA

Potresti ricevere questo errore quando il nucleo non è in grado di convalidare la configurazione distribuita per il componente. Controlla il registro per i dettagli degli errori, quindi riprova a eseguire la distribuzione.

Errore del dispositivo

IO_WRITE_ERROR

Potresti ricevere questo errore durante la scrittura su un file. Controlla il registro per i dettagli.

IO_READ_ERROR

Potresti ricevere questo errore durante la lettura da un file. Controlla il registro per i dettagli.

SPAZIO SU DISCO CRITICO

È possibile che venga visualizzato questo errore quando lo spazio su disco non è sufficiente per completare una richiesta di distribuzione. Devi disporre di almeno 20 Mb di spazio disponibile o sufficiente per contenere un artefatto più grande. Libera spazio su disco e riprova a eseguire la distribuzione.

IO_FILE_ATTRIBUTE_ERROR

Potresti ricevere questo errore quando la dimensione del file esistente non può essere recuperata dal file system. Controlla il registro per i dettagli.

SET_PERMISSION_ERROR

Potresti ricevere questo errore quando non è possibile impostare le autorizzazioni su un artefatto o una directory di artefatti scaricati. Controlla il registro per i dettagli.

IO_UNZIP_ERROR

Potresti ricevere questo errore quando un artefatto non può essere decompresso. Controlla il registro per i dettagli.

RECIPE_LOCALE NON TROVATA

Potresti ricevere questo errore quando non è stata trovata la copia locale di un file di ricetta. Prova a eseguire nuovamente la distribuzione.

LOCAL_RECIPE_DANNEGGIATA

Potresti ricevere questo errore quando la copia locale della ricetta è cambiata da quando è stata scaricata. Elimina la copia esistente della ricetta e riprova a eseguire la distribuzione.

RECIPE_METADATI LOCALI NON TROVATI

Potresti ricevere questo errore quando la copia locale del file di metadati della ricetta non è stata trovata. Prova a eseguire nuovamente la distribuzione.

LAUNCH_DIRECTORY_DANNEGGIATO

Potresti ricevere questo errore quando la directory viene utilizzata per avviare il nucleo di Greengrass (`/greengrass/v2/alts/current`) è stato modificato dall'ultima volta che il nucleo è stato avviato. Riavvia il nucleo e quindi riprova a dispiegarlo.

ALGORITMO_HASHING NON DISPONIBILE

Potresti ricevere questo errore quando la distribuzione Java del dispositivo non supporta l'algoritmo di hashing richiesto o quando l'algoritmo hash specificato nella ricetta di un componente non è valido.

DEVICE_CONFIG_NOT_VALID_PER_ARTIFACT_DOWNLOAD

Potresti ricevere questo errore quando c'è un errore nella configurazione del dispositivo che ha impedito alla distribuzione di scaricare l'artefatto da Amazon S3 o dal cloud Greengrass. Controlla il registro per un errore di configurazione specifico, quindi riprova la distribuzione.

Errore di dipendenza

ERRORE DOCKER

Potresti ricevere questo errore quando estrai un'immagine Docker. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

DOCKER_SERVICE_NON DISPONIBILE

Potresti ricevere questo errore quando Greengrass non riesce ad accedere al registro Docker. Controlla nel registro la presenza di un errore specifico, quindi riprova a eseguire la distribuzione.

DOCKER_LOGIN_ERROR

Potresti ricevere questo errore quando si verifica un errore imprevisto durante l'accesso a Docker. Controlla nel registro la presenza di un errore specifico, quindi riprova a eseguire la distribuzione.

DOCKER_PULL_ERROR

Potresti ricevere questo errore quando si verifica un errore imprevisto durante l'estrazione di un'immagine Docker dal registro. Controlla nel registro la presenza di un errore specifico, quindi riprova a eseguire la distribuzione.

DOCKER_IMAGE_NON VALIDA

Potresti ricevere questo errore quando l'immagine Docker richiesta non esiste. Controlla nel registro la presenza di un errore specifico e riprova a eseguire la distribuzione.

DOCKER_IMAGE_QUERY_ERROR

Potresti ricevere questo errore quando si verifica un errore imprevisto durante la richiesta di immagini disponibili a Docker. Controlla nel registro l'errore specifico e riprova a eseguire la distribuzione.

ERRORE S3

Potresti ricevere questo errore durante il download di un artefatto di Amazon S3. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

S3_RISORSA_NON TROVATA

Potresti ricevere questo errore quando un'operazione Amazon S3 restituisce un errore 404. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

S3_RICHIESTA ERRATA

Potresti ricevere questo errore quando un'operazione Amazon S3 restituisce un errore 400. Controlla nel registro la presenza di un errore specifico e riprova a eseguire la richiesta.

Errore HTTP

HTTP_REQUEST_ERROR

Potresti ricevere questo errore quando si verifica un errore durante la richiesta HTTP. Controlla il registro per l'errore specifico.

DOWNLOAD_DEPLOYMENT_DOCUMENT_ERROR

È possibile che questo errore venga visualizzato quando si verifica un errore HTTP durante il download del documento di distribuzione. Controlla il registro per l'errore HTTP specifico.

GET_GREENGRASS_ARTIFACT_SIZE_ERROR

Potresti ricevere questo errore quando si verifica un errore HTTP durante la rilevazione delle dimensioni di un elemento pubblico. Controlla il registro per l'errore HTTP specifico.

DOWNLOAD_GREENGRASS_ARTIFACT_ERROR

Potresti ricevere questo errore quando si verifica un errore HTTP durante il download di un componente pubblico. Controlla il registro per l'errore HTTP specifico.

Errore di rete

ERRORE_DI RETE

Potresti ricevere questo errore quando si verifica un problema di connessione durante una distribuzione. Verificare la connessione del dispositivo a Internet e riprovare a eseguire l'installazione.

Errore Nucleus

RICHIESTA_ERRATA

Potresti ricevere questo errore quandoAWSl'operazione cloud restituisce un errore 400. Controlla il registro per vedere quale API ha causato l'errore, quindi controlla la pagina di aggiornamento

del software nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

VERSIONE DEL NUCLEO NON TROVATA

Potresti ricevere questo errore quando un dispositivo principale non riesce a trovare la versione del nucleo attivo. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

ERRORE DI RIAVVIO DEL NUCLEO

Potresti ricevere questo errore quando il nucleo non si riavvia durante una distribuzione che richiede il riavvio del nucleo. Controlla il log del loader per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus, oppure contattaAWS Support.

COMPONENT_INSTALLATO_NOT_FOUND

Potresti ricevere questo errore quando il nucleo non riesce a localizzare un componente installato. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

DOCUMENTO_DI DISTRIBUZIONE NON VALIDO

È possibile che venga visualizzato questo errore quando il dispositivo riceve un documento di distribuzione non valido. Controlla eventuali codici di errore aggiuntivi o controlla il registro per vedere cosa ha causato l'errore.

RICHIESTA_DI_DISTRIBUZIONE VUOTA

È possibile che venga visualizzato questo errore quando un dispositivo riceve una richiesta di distribuzione vuota. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

DEPLOYMENT_DOCUMENT_PARSE_ERROR

È possibile che venga visualizzato questo errore quando il formato della richiesta di distribuzione non corrisponde al formato previsto. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

METADATI DEI COMPONENTI NON VALIDI NELLA DISTRIBUZIONE

È possibile che venga visualizzato questo errore quando la richiesta di distribuzione contiene metadati dei componenti non validi. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

LAUNCH_DIRECTORY_DANNEGGIATO

Potresti ricevere questo errore quando sposti un dispositivo Greengrass da un gruppo di cose a un altro e quindi di nuovo al gruppo originale con distribuzioni che richiedono il riavvio di Greengrass. Per risolvere l'errore, ricreate la directory di avvio di Greengrass sul dispositivo.

Per ulteriori informazioni, consulta Error: com.aws.greengrass.deployment.exceptions.DeploymentException: Unable to process deployment. Greengrass launch directory is not set up or Greengrass is not set up as a system service.

Errore del server

ERRORE_SERVER

Potresti ricevere questo errore quandoAWSl'operazione del servizio restituisce un errore 500 perché il servizio non è in grado di elaborare la richiesta in questo momento. Riprova la distribuzione in un secondo momento.

ERRORE DEL SERVER S3

Potresti ricevere questo errore quando un'operazione Amazon S3 restituisce un errore 500. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

Errore del servizio cloud

RESOLVE_COMPONENT_CANDIDATI_BAD_RESPONSE

Potresti ricevere questo errore quando il servizio cloud Greengrass invia una risposta incompatibile al`ResolveComponentCandidates`operazione. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

DIMENSIONE DEL DOCUMENTO DI DISTRIBUZIONE SUPERATA

È possibile che venga visualizzato questo errore quando il documento di distribuzione richiesto supera la quota di dimensione massima. Riduci le dimensioni del documento di distribuzione e riprova a eseguire la distribuzione.

ERBA VERDE DEL MANUFATTO DI DIMENSIONI NON TROVATE

Potresti ricevere questo errore quando Greengrass non riesce a ottenere le dimensioni di un artefatto di un componente pubblico. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

DOCUMENTO_DI DISTRIBUZIONE NON VALIDO

È possibile che venga visualizzato questo errore quando il dispositivo riceve un documento di distribuzione non valido. Controlla eventuali codici di errore aggiuntivi o controlla il registro per vedere cosa ha causato l'errore.

RICHIESTA_DI_DISTRIBUZIONE VUOTA

È possibile che venga visualizzato questo errore quando un dispositivo riceve una richiesta di distribuzione vuota. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

DEPLOYMENT_DOCUMENT_PARSE_ERROR

È possibile che venga visualizzato questo errore quando il formato della richiesta di distribuzione non corrisponde al formato previsto. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

METADATI DEI COMPONENTI NON VALIDI NELLA DISTRIBUZIONE

È possibile che venga visualizzato questo errore quando la richiesta di distribuzione contiene metadati dei componenti non validi. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

Errori generici

Questi errori generici non hanno un tipo di errore associato.

DISTRIBUZIONE INTERROTTA

Potresti ricevere questo errore quando una distribuzione non può essere completata a causa di un arresto del nucleo o di un altro evento esterno. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

ARTIFACT_DOWNLOAD_ERROR

Potresti ricevere questo errore quando si verifica un problema durante il download di un artefatto. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

NESSUNA VERSIONE DEI COMPONENTI DISPONIBILE

Potresti ricevere questo errore quando la versione di un componente non esiste nel cloud o localmente o se c'è un conflitto di risoluzione delle dipendenze. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

ERRORE DI CARICAMENTO DEL PACCHETTO DI COMPONENTI

Potresti ricevere questo errore quando si verifica un errore durante l'elaborazione degli artefatti scaricati. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

ERRORE CLOUD_API

Potresti ricevere questo errore quando si verifica un errore durante la chiamata aAWSAPI di servizio. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

IO_ERROR

È possibile che questo errore venga visualizzato quando si verifica un errore di I/O durante una distribuzione. Controlla eventuali codici di errore o registri aggiuntivi per i dettagli.

ERRORE DI AGGIORNAMENTO DEI COMPONENTI

Potresti ricevere questo errore quando un componente non viene aggiornato durante una distribuzione. Controlla eventuali codici di errore aggiuntivi o controlla il registro per vedere cosa ha causato l'errore.

Errore sconosciuto

FALLIMENTO_DI INSTALLAZIONE

Potresti ricevere questo errore quando una distribuzione fallisce perché è stata generata un'eccezione deselezionata. Controlla il registro per vedere cosa ha causato l'errore, quindi

controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

TIPO_DIPLOYMENT_NOT_VALID

Potresti ricevere questo errore quando il tipo di distribuzione non è valido. Controlla il registro per vedere cosa ha causato l'errore, quindi controlla la pagina di aggiornamento del software Nucleus per vedere se il problema è stato corretto in una versione successiva del nucleus oppure contattaAWS Support.

Codici di stato dettagliati dei componenti

Usa i codici di stato e le soluzioni in queste sezioni per risolvere i problemi con i componenti quando usi la versione 2.8.0 o successiva di Greengrass nucleus.

Molti degli stati di questo argomento riportano informazioni aggiuntive nei registri AWS IoT Greengrass principali. Questi registri vengono archiviati nel file system locale del dispositivo principale. Esistono registri per ogni singolo componente. Per informazioni sull'accesso ai registri, vedere. Accedere ai log del file system

ERRORE DI INSTALLAZIONE

È possibile che venga visualizzato quando si verifica un errore durante l'esecuzione di uno script di installazione. Il codice di errore è riportato nel registro dei componenti. Verifica la presenza di errori nello script di installazione e distribuisci nuovamente il componente.

INSTALL_CONFIG_NOT_VALID

Potresti ricevere questo errore quando l'installazione di un componente non può essere completata perché la `Install` sezione della ricetta non è valida. Controlla la sezione di installazione della tua ricetta per verificare la presenza di errori e riprova a eseguire la distribuzione.

INSTALL_IO_ERROR

Questo problema può verificarsi quando si verifica un errore di I/O durante l'installazione di un componente. Controlla il log degli errori del componente per i dettagli sull'errore.

INSTALL_MISSING_DEFAULT_RUNWITH

Potresti ricevere questo errore quando non AWS IoT Greengrass riesci a determinare l'utente o il gruppo da utilizzare durante l'installazione di un componente. Verifica che la `runWith` sezione della ricetta di installazione includa un utente o un gruppo valido.

TIMEOUT DI INSTALLAZIONE

È possibile che venga visualizzato questo errore quando lo script di installazione non è stato completato entro il periodo di timeout configurato. Aumenta il `Timeout` periodo specificato nella `Install` sezione della ricetta o modifica lo script di installazione per terminare entro il timeout configurato.

ERRORE DI AVVIO

È possibile che venga visualizzato quando si verifica un errore durante l'esecuzione di uno script di avvio. Il codice di errore è riportato nel registro dei componenti. Verifica la presenza di errori nello script di installazione e distribuisci nuovamente il componente.

STARTUP_CONFIG_NON VALIDO

Potresti ricevere questo errore quando l'installazione di un componente non può essere completata perché la `Startup` sezione della ricetta non è valida. Controlla la sezione di avvio della tua ricetta per verificare la presenza di errori e riprova a eseguire la distribuzione.

ERRORE IO_DI AVVIO

Questo problema può verificarsi quando si verifica un errore di I/O durante l'avvio di un componente. Controlla il log degli errori del componente per i dettagli sull'errore.

STARTUP_MISSING_DEFAULT_RUNWITH

Potresti ricevere questo errore quando non AWS IoT Greengrass riesci a determinare l'utente o il gruppo da utilizzare durante l'esecuzione di un componente. Verifica che la `runWith` sezione della tua ricetta di avvio includa un utente o un gruppo valido.

TIMEOUT DI AVVIO

È possibile che venga visualizzato questo errore quando lo script di avvio non è stato completato entro il periodo di timeout configurato. Aumenta il `Timeout` periodo specificato nella `Startup` sezione della ricetta o modifica lo script di avvio per terminare entro il timeout configurato.

RUN_ERROR

È possibile che venga visualizzato quando si verifica un errore durante l'esecuzione di uno script di componente. Il codice di errore è riportato nel registro dei componenti. Verifica la presenza di errori nello script di esecuzione e distribuisci nuovamente il componente.

RUN_MISSING_DEFAULT_RUNWITH

Potresti ricevere questo errore quando non AWS IoT Greengrass riesci a determinare l'utente o il gruppo da utilizzare durante l'esecuzione di un componente. Verifica che la `runWith` sezione della tua ricetta di corsa includa un utente o un gruppo valido.

ESEGUI CONFIG_NOT_VALID

Potresti ricevere questo errore quando un componente non può essere eseguito perché la `Run` sezione della ricetta non è valida. Controlla la sezione di esecuzione della tua ricetta per verificare la presenza di errori e riprova a eseguire la distribuzione.

RUN_IO_ERROR

È possibile che venga visualizzato quando si è verificato un errore di I/O mentre il componente è in esecuzione. Controlla il log degli errori del componente per i dettagli sull'errore.

RUN_TIMEOUT

È possibile che venga visualizzato questo errore quando lo script di esecuzione non è stato completato entro il periodo di timeout configurato. Aumenta il `Timeout` periodo specificato nella `Run` sezione della ricetta o modifica lo script di esecuzione per terminare entro il timeout configurato.

SHUTDOWN_ERROR

È possibile che venga visualizzato quando si verifica un errore durante la chiusura dello script di un componente. Il codice di errore è riportato nel registro dei componenti. Verifica la presenza di errori nello script di spegnimento e distribuisci nuovamente il componente.

TIMEOUT DI SPEGNIMENTO

È possibile che venga visualizzato questo errore quando lo script di spegnimento non è stato completato entro il periodo di timeout configurato. Aumenta il `Timeout` periodo specificato nella `Shutdown` sezione della ricetta o modifica lo script di esecuzione per terminare entro il timeout configurato.

Tagging delle risorse AWS IoT Greengrass Version 2.

Con i tag, puoi organizzare e gestire le risorse in AWS IoT Greengrass. Puoi utilizzare i tag per assegnare metadati alle tue risorse e puoi utilizzare i tag nelle politiche IAM per definire l'accesso condizionale alle tue risorse.

> **Note**
>
> Attualmente, i tag delle risorse Greengrass non sono supportati per i gruppi di fatturazione o i report di allocazione dei costi di AWS IoT.

Utilizzo dei tag in AWS IoT Greengrass V2

Puoi utilizzare i tag per classificare le risorse AWS IoT Greengrass in base a scopo, proprietario, ambiente o qualsiasi altra classificazione per il caso d'uso. In presenza di un numero elevato di risorse dello stesso tipo, i tag consentono di identificare più facilmente una risorsa specifica.

Ogni tag è composto da una chiave e da un valore opzionale, entrambi personalizzabili. Ad esempio, puoi definire un set di tag per i tuoi dispositivi principali che consentono di monitorare i clienti che possiedono i dispositivi. Ti consigliamo di creare un set di chiavi di tag in grado di soddisfare i requisiti di ciascun tipo di risorsa. Utilizzando un set di chiavi di tag coerenti, puoi gestire più facilmente le risorse.

Etichetta conAWS Management Console

Tag Editor (Editor di tag) nella AWS Management Console fornisce un modo centrale, unificato per creare e gestire i tag per le risorse di tutti i servizi AWS. Per ulteriori informazioni, consulta Editor di tag nella Guida per l'utente di AWS Resource Groups.

Tagga con l'AWS IoT Greengrass V2API

È possibile utilizzare l'AWS IoT Greengrass V2API anche per lavorare con i tag. Prima di creare i tag, tenere presente le limitazioni relative al tagging. Per ulteriori informazioni, consulta la sezione relativa alle convenzioni di denominazione e utilizzo dei tag nella Riferimenti generali di AWS.

- Per aggiungere i tag durante la creazione di una risorsa, definirli nella proprietà `tags` della risorsa.

- Per aggiungere tag a una risorsa esistente o per aggiornare i valori dei tag, usa l'TagResourceoperazione.
- Per rimuovere i tag da una risorsa, utilizza l'UntagResourceoperazione.
- Per recuperare i tag associati a una risorsa, usa l'ListTagsForResourceoperazione o descrivi la risorsa e controllane la`tags` proprietà.

La tabella seguente elenca le risorse che puoi etichettare utilizzando l'AWS IoT Greengrass V2API e le relative`CreateGet` operazioni`Describe` e/o.

Risorse AWS IoT Greengrass V2 compatibili con l'applicazione di tag

Risorsa	Creare l'operazione	Descrivi o ottieni l'operazione
Dispositivo principale	Nessuna. Esegui il softwareA WS IoT Greengrass Core su un dispositivo per creare un dispositivo principale.	GetCoreDevice
Componente	CreateComponentVersion	DescribeComponent, GetComponent
Distribuzione	CreateDeployment	GetDeployment

Utilizza le operazioni seguenti per elencare e gestire i tag per le risorse che supportano il tagging:

- TagResource— Aggiunge tag a una risorsa o aggiorna il valore di un tag esistente.
- ListTagsForResource— Elenca i tag di una risorsa.
- UntagResource— Rimuove i tag da una risorsa.

Puoi aggiungere o rimuovere tag per una risorsa in qualsiasi momento. Per modificare il valore di una chiave del tag, aggiungere un tag alla risorsa che definisca la stessa chiave e il nuovo valore. Il nuovo valore sostituisce il valore precedente. Puoi impostare il valore su una stringa vuota, ma non su null.

Se elimini una risorsa, verranno eliminati anche tutti i tag associati alla risorsa.

Utilizzo dei tag con policy IAM

Nelle tue policy IAM, puoi utilizzare i tag delle risorse per controllare l'accesso e le autorizzazioni degli utenti e le autorizzazioni. Ad esempio, le policy possono consentire agli utenti di creare solo le risorse con un determinato tag. Le policy possono anche limitare gli utenti nella creazione o nella modifica di risorse con determinati tag.

> **Note**
>
> Se utilizzi tag per consentire o rifiutare agli utenti di accedere alle risorse, devi negare agli utenti la possibilità di aggiungere o rimuovere tali tag dalle stesse risorse. In caso contrario, un utente potrebbe eludere le restrizioni e ottenere l'accesso a una risorsa modificandone i tag.

È possibile utilizzare le seguenti chiavi e valori di contesto delle condizioni nell'`Condition`elemento, chiamato anche`Condition` blocco, di una dichiarazione politica.

`greengrassv2:ResourceTag/`*`tag-key`*`: `*`tag-value`*

Consentire o negare agli utenti operazioni su risorse con tag specifici.

`aws:RequestTag/`*`tag-key`*`: `*`tag-value`*

Richiede che venga utilizzato o meno un tag specifico durante la creazione o la modifica di una risorsa etichettabile.

`aws:TagKeys: [`*`tag-key`*`, ...]`

Richiede che venga utilizzato o meno un set specifico di chiavi di tag durante la creazione o la modifica di una risorsa etichettabile.

> **Note**
>
> Le chiavi e i valori del contesto della condizione in una politica IAM si applicano solo alle azioni che hanno una risorsa etichettabile come parametro obbligatorio. Ad esempio, puoi impostare l'accesso condizionale basato su tag per ListCoreDevices.

Per ulteriori informazioni, consulta Controllare l'accesso alleAWS risorse utilizzando i tag delle risorse e il riferimento alla politica IAM JSON nella Guida per l'utente IAM.

Creazione di risorse AWS IoT Greengrass con AWS CloudFormation

AWS IoT Greengrass è integrato con AWS CloudFormation, un servizio che ti consente di modellare e configurare le tue risorse AWS in modo da dedicare meno tempo alla creazione e alla gestione delle risorse e dell'infrastruttura. Crei un modello che descrive tutte leAWSrisorse desiderate (ad esempio versioni e distribuzioni dei componenti) eAWS CloudFormationfornisce e configura tali risorse per te.

Quando usi AWS CloudFormation, puoi riutilizzare il modello per configurare le risorse AWS IoT Greengrass in modo coerente e continuo. Basta descrivere le risorse una volta sola, dopodiché si può effettuare il provisioning di tali risorse quante volte si vuole in più Account AWS e regioni.

AWS IoT Greengrass e modelli AWS CloudFormation

Per eseguire l'assegnazione e la configurazione delle risorse per AWS IoT Greengrass e i servizi correlati, devi conoscere i modelli AWS CloudFormation. I modelli sono file di testo formattati in JSON o YAML. Questi modelli descrivono le risorse di cui intendi effettuare il provisioning negli stack AWS CloudFormation. Se non hai familiarità con JSON o YAML, puoi usare AWS CloudFormation Designer per iniziare a utilizzare i modelli AWS CloudFormation. Per ulteriori informazioni, consulta Che cos'è AWS CloudFormation Designer? nella Guida per l'utente di AWS CloudFormation.

AWS IoT Greengrasssupporta la creazione di versioni e distribuzioni di componenti inAWS CloudFormation. Per ulteriori informazioni, inclusi esempi di modelli JSON e YAML per le versioni e le distribuzioni dei componenti, consultaAWS IoT GreengrassInformazioni sul tipo di risorsanelAWS CloudFormationGuida per l'utente di.

ComponentVersion Esempio di modello

Di seguito è riportato il modello YAML per una versione di un componente semplice. La ricetta JSON include interruzioni di riga per migliorare la leggibilità.

```
Parameters:
  ComponentVersion:
    Type: String
Resources:
  TestSimpleComponentVersion:
```

```
    Type: AWS::GreengrassV2::ComponentVersion
    Properties:
      InlineRecipe: !Sub
        - "{\n
            \"RecipeFormatVersion\": \"2020-01-25\",\n
            \"ComponentName\": \"component1\",\n
            \"ComponentVersion\": \"${ComponentVersion}\",\n
            \"ComponentType\": \"aws.greengrass.generic\",\n
            \"ComponentDescription\": \"This\",\n
            \"ComponentPublisher\": \"You\",\n
            \"Manifests\": [\n
            {\n
              \"Platform\": {\n
              \"os\": \"darwin\"\n
            },\n
            \"Lifecycle\": {},\n
            \"Artifacts\": []\n
           },\n
           {\n
             \"Lifecycle\": {},\n
             \"Artifacts\": []\n
           }\n
          ],\n
          \"Lifecycle\": {\n
          \"install\": {\n
            \"script\": \"yuminstallpython\"\n
          }\n
         }\n
        }"
        - { ComponentVersion: !Ref ComponentVersion }
```

Esempi di modello di distribuzione

Di seguito è riportato un file YAML che definisce un modello semplice per una distribuzione.

```
Parameters:
  ComponentVersion:
    Type: String
  TargetArn:
    Type: String
Resources:
  TestDeployment:
    Type: AWS::GreengrassV2::Deployment
```

```
    Properties:
      Components:
        component1:
          ComponentVersion: !Ref ComponentVersion
      TargetArn: !Ref TargetArn
      DeploymentName: CloudFormationIntegrationTest
      DeploymentPolicies:
        FailureHandlingPolicy: DO_NOTHING
        ComponentUpdatePolicy:
          TimeoutInSeconds: 5000
          Action: SKIP_NOTIFY_COMPONENTS
        ConfigurationValidationPolicy:
          TimeoutInSeconds: 30000
Outputs:
  TestDeploymentArn:
    Value: !Sub
      - arn:${AWS::Partition}:greengrass:${AWS::Region}:${AWS::AccountId}:deployments:
${DeploymentId}
      - DeploymentId: !GetAtt TestDeployment.DeploymentId
```

Ulteriori informazioni su AWS CloudFormation

Per ulteriori informazioni su AWS CloudFormation, consulta le seguenti risorse:

- AWS CloudFormation
- Guida per l'utente di AWS CloudFormation
- Documentazione di riferimento dell'API AWS CloudFormation
- Guida per l'utente dell'interfaccia a riga di comando di AWS CloudFormation

Software AWS IoT Greengrass Core open source

L'AWS IoT Greengrass Version 2edge runtime (nucleus) e gli altri componenti del software AWS IoT Greengrass Core sono open source. Ciò significa che puoi rivedere il codice per risolvere i problemi di interazione con le tue applicazioni. È inoltre possibile personalizzare ed estendere il software AWS IoT Greengrass Core per soddisfare esigenze software e hardware specifiche.

Per informazioni sui repository open source per il software AWS IoT Greengrass Core, consulta l'organizzazione aws-greengrass su. GitHub L'uso del software open source è regolato dalla licenza open source nel repository corrispondente. GitHub

L'utilizzo del software e dei componenti AWS IoT Greengrass Core non soggetti a una licenza open source è regolato dalla licenza software AWS Greengrass Core.

Cronologia dei documenti per la AWS IoT Greengrass V2 Developer Guide

La tabella seguente descrive la documentazione per questa versione di AWS IoT Greengrass Version 2.

- Versione API: 2020-11-30

Modifica	Descrizione	Data
AWS IoT Device Tester v4.9.3 con GGV2Q v2.5.3 rilasciato	È disponibile la versione AWS IoT Greengrass 4.9.3 di IDT per V2. Questa versione include la suite di qualifica zione AWS IoT Greengrass V2 (GGV2Q) v2.5.3 e supporta le versioni 2.12.0, 2.11.0, 2.10.0, 2.9.5 di Greengrass nucleus.	5 aprile 2024
Rilasciata la CLI Greengrass v2.12.4	Il componente Greengrass CLI v2.12.4 è disponibile.	2 aprile 2024
AWS IoT Greengrass Aggiornamento del software Core v2.12.4	Questa versione fornisce la versione 2.12.4 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	2 aprile 2024
Rilasciato Shadow Manager v2.3.7	Shadow manager v2.3.7 è disponibile. Questa versione corregge un problema a causa del quale shadow manager registra periodicamente un `NullPointerException` errore durante la sincroniz	27 marzo 2024

	zazione di uno shadow manager.	
Rilasciato il broker Moquette MQTT 3.1.1 v2.3.6	È disponibile il component e broker Moquette MQTT 3.1.1 v2.3.6. Questa versione include correzioni di bug e miglioramenti generali.	27 marzo 2024
Rilasciata la console di debug locale v2.4.2	È disponibile il component e v2.4.2 della console di debug locale. Questa versione include correzioni di bug e miglioramenti generali.	27 marzo 2024
Lambda Manager v2.3.3 rilasciato	È disponibile il component e Lambda Manager v2.3.3. Questa versione include correzioni di bug e miglioram enti generali.	27 marzo 2024
Rilasciato il rilevatore IP v2.1.9	È disponibile il componente del rilevatore IP v2.1.9. Questa versione regola la fase di acquisizione dell'IP in modo da inviare i log solo a livello di registro di debug.	27 marzo 2024

AWS IoT Rilasciato il plugin per il provisioning della flotta v1.2.1	AWS IoT è disponibile il plugin per il provisioning della flotta v1.2.1. Questa versione risolve un problema a causa del quale il plug-in di provisioning della flotta è offline durante l'avvio di Greengrass nucleus. Il plug-in di provisioning della flotta ora riprova a tempo indeterminato le chiamate di connessione MQTT.	27 marzo 2024
AWS IoT Greengrass Aggiornamento del software Core v2.12.3	Questa versione fornisce la versione 2.12.3 del componente Greengrass nucleus e aggiorna i componenti forniti. AWS	27 marzo 2024
Rilasciata la versione 2.12.3 della CLI Greengrass	Il componente Greengrass CLI v2.12.3 è disponibile.	25 marzo 2024
AWS IoT Device Tester è stata rilasciata la versione 4.9.2 con GGV2Q v2.5.2	È disponibile la versione AWS IoT Greengrass 4.9.2 di IDT per V2. Questa versione include la suite di qualificazione AWS IoT Greengrass V2 (GGV2Q) v2.5.2 e supporta le versioni 2.12.0, 2.11.0, 2.10.0, 2.9.5 di Greengrass nucleus.	18 marzo 2024
Rilasciato l'agente edge Lookout for Vision v1.2.0	È disponibile l'agente Lookout for Vision edge v1.2.0.	11 marzo 2024

AWS IoT Greengrass Aggiornamento del software Core v2.12.2	Questa versione fornisce la versione 2.12.2 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	15 febbraio 2024
Rilasciato Shadow Manager v2.3.6	Shadow manager v2.3.6 è disponibile. Questa versione corregge un problema per cui le proprietà shadow che vengono eliminate tramite Cloud AWS aggiornam enti mentre il dispositivo è offline continuano a esistere nell'ombra locale dopo il ripristino della connettività.	14 febbraio 2024
Lambda Launcher v2.0.13 rilasciato	È disponibile la versione 2.0.13 del componente di avvio Lambda. Questa versione include correzioni di bug e miglioramenti generali.	14 febbraio 2024
Rilasciato Disk spooler v1.0.3	È disponibile il componente Disk spooler v1.0.3. Questa versione migliora le prestazioni riutilizzando le connessioni al database.	14 febbraio 2024
Rilasciato l'agente edge Lookout for Vision v1.1.9	È disponibile l'agente Lookout for Vision edge v1.1.9.	17 gennaio 2024

Kit di sviluppo Greengrass CLI v1.6.2	È disponibile la versione 1.6.2 del Greengrass Developme nt Kit CLI. Questa versione corregge un problema in cui Windows gradlew.bat non funziona a causa del percorso relativo. Questa versione contiene anche ulteriori miglioramenti.	16 gennaio 2024
Nuovi eventi CloudTrail relativi ai dati	Ora puoi registrare gli eventi AWS CloudTrail relativi ai dati per ottenere informazioni sulle operazioni relative alle risorse, ad esempio l'acquisi zione di un componente o la configurazione di una distribuz ione. Usa questi eventi per ottenere informazioni dettaglia te sul funzionamento dei tuoi dispositivi Greengrass.	20 dicembre 2023
Rilasciato l'agente edge Lookout for Vision v1.1.8	È disponibile l'agente Lookout for Vision edge v1.1.8.	12 dicembre 2023
È stato rilasciato Stream Manager v2.1.12	Stream manager v2.1.12 è ora disponibile. Questa versione modifica l'ordine utilizzato da Greengrass per selezionare un set di credenziali per AWS le chiamate di servizio.	8 dicembre 2023
Rilasciato il bridge MQTT v2.3.1	È disponibile il bridge MQTT v2.3.1. Questa versione risolve un problema raro in cui il client MQTT locale entra in un ciclo di disconnessione.	8 dicembre 2023

Rilasciato Disk spooler v1.0.2	È disponibile il component e Disk spooler v1.0.2. Questa versione corregge un problema per cui il campo del formato dei messaggi MQTT non viene mantenuto in alcuni casi.	8 dicembre 2023
È stato rilasciato il component e di autenticazione del dispositivo client v2.4.5	È disponibile il componente di autenticazione del dispositivo client v2.4.5. Questa versione aggiunge il supporto per i caratteri jolly alla fine dei nomi degli oggetti in una regola di selezione e corregge un problema per cui i certificati non vengono aggiornati con nuove informazioni di connettiv ità in alcuni casi.	8 dicembre 2023
AWS IoT Greengrass Aggiornamento del software Core v2.12.1	Questa versione fornisce la versione 2.12.1 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	8 dicembre 2023
Kit di sviluppo Greengrass CLI v1.6.1	È disponibile la versione 1.6.1 del Greengrass Developme nt Kit CLI. Questa versione contiene correzioni di bug e miglioramenti.	6 dicembre 2023

Convalida della ricetta	È stata aggiunta una funzionalità di convalida della ricetta che convaliderà la ricetta di un componente durante la creazione di una versione del componente.	16 novembre 2023
Componenti supportati da Publisher	AWS IoT Greengrass ora offre componenti supportati da Publisher. Questi componenti sono sviluppati, offerti e sottoposti a manutenzione da fornitori terzi.	16 novembre 2023
Rilasciato Greengrass Testing Framework v1.2.0	Greengrass Testing Framework v1.2.0 è disponibile.	15 novembre 2023

Kit di sviluppo Greengrass CLI v1.6.0	È disponibile la versione 1.6.0 del Greengrass Developme nt Kit CLI. Questa versione aggiunge un controllo di convalida della ricetta rispetto allo schema della ricetta Greengrass durante `component build` i `component publish` comandi and. Questo aggiornamento aiuta gli sviluppatori a identificare i problemi risolvibili all'interno delle ricette dei componenti nelle prime fasi del processo di creazione dei component i. Questa versione aggiunge anche una suite di test di confidenza al modello che può essere rimossa dal comando. `test-e2e init` Questa suite di test di fiducia include otto test generici che possono essere utilizzati ed estesi per soddisfare le esigenze di test dei componenti di base.	15 novembre 2023
AWS IoT Device Tester v4.9.1 supporta la versione 2.12.0 di Greengrass nucleus	La versione 4.9.1 di IDT per AWS IoT Greengrass V2 ora supporta la versione 2.12.0 di Greengrass nucleus.	7 novembre 2023

AWS IoT Greengrass Aggiornamento del software Core v2.12.0	Questa versione fornisce la versione 2.12.0 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	7 novembre 2023
Utilizza un dispositivo core Greengrass in VPC	È possibile utilizzare un dispositivo core Greengras s in VPC. Questa funzional ità consente di eseguire implementazioni in VPC senza accesso pubblico a Internet.	3 novembre 2023
Rilasciata la CLI Greengrass v2.12.0	Il componente Greengrass CLI v2.12.0 è disponibile.	30 ottobre 2023
È stato rilasciato lo Stream Manager v2.1.10	Stream manager v2.1.10 è ora disponibile. Questa versione corregge un problema per cui la configurazione del proxy HTTPS non considera attendibile la catena di certifica ti Greengrass CA.	26 ottobre 2023
Lambda Launcher v2.0.12 rilasciato	È disponibile la versione 2.0.12 del componente di avvio Lambda. Questa versione corregge un problema a causa del quale il programma di avvio Lambda poteva generare un errore se il processo precedente non veniva interrotto correttam ente.	26 ottobre 2023

Kit di sviluppo Greengrass CLI v1.5.0	È disponibile la versione 1.5.0 del Greengrass Developme nt Kit CLI. Questa versione aggiorna i modelli riconosci uti dall'opzione `excludes` build quando è disponibile. `build_system zip` Questa versione ora riconoscerà i pattern globulari che corrispon dono ai nomi dei percorsi in base ai loro caratteri jolly. Ciò consente di specificare in modo personalizzato le directory da cui escludere.	26 ottobre 2023
Rilasciato l'agente Lookout for Vision edge v1.1.7	È disponibile l'agente Lookout for Vision edge v1.1.7.	24 ottobre 2023
È stato rilasciato Shadow Manager v2.3.4	Shadow manager v2.3.4 è disponibile. Questa versione aggiunge il supporto per i documenti dello stato ombra nulli e vuoti.	18 ottobre 2023
Rilasciato Log Manager v2.3.6	Il componente Log Manager v2.3.6 è disponibile.	18 ottobre 2023
È stata rilasciata la console di debug locale v2.4.0	È disponibile il componente v2.4.0 della console di debug locale.	18 ottobre 2023
Lambda Manager v2.3.1 rilasciato	È disponibile il componente Lambda Manager v2.3.1.	18 ottobre 2023
Rilasciata la versione 2.11.3 della CLI Greengrass	Il componente Greengrass CLI v2.11.3 è disponibile.	18 ottobre 2023

AWS IoT Greengrass Aggiornamento del software Core v2.11.3	Questa versione fornisce la versione 2.11.3 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	18 ottobre 2023
Rilasciato Secure Tunneling v1.0.17	È disponibile il tunneling sicuro v1.0.17.	4 ottobre 2023
Kit di sviluppo Greengrass CLI v1.4.0	È disponibile la versione 1.4.0 del Greengrass Developme nt Kit CLI. Questa versione aggiunge un nuovo `config` comando che avvia un prompt interattivo per modificare i campi all'interno di un file di configurazione GDK esistente . Questa versione modifica anche `gdk component publish` i comandi `gdk component build` and per verificare che la dimensione della ricetta rientri nei requisiti di Greengrass (<=16000 byte) prima di procedere.	2 ottobre 2023
È stato rilasciato il broker Moquette MQTT 3.1.1 v2.3.5	È disponibile il component e broker Moquette MQTT 3.1.1 v2.3.5. Questa versione aggiorna Moquette alla versione 0.17.	28 settembre 2023
Rilasciato il bridge MQTT v2.3.0	È disponibile il bridge MQTT v2.3.0. Questa versione aggiunge il supporto MQTT 5 per il collegamento tra AWS IoT Core sorgenti MQTT locali.	28 settembre 2023

Rilasciato l'agente edge Lookout for Vision v1.1.6	È disponibile l'agente Lookout for Vision edge v1.1.6.	27 settembre 2023
Lambda Manager v2.3.0 rilasciato	È disponibile il componente Lambda Manager v2.3.0.	15 settembre 2023
Lambda Launcher v2.0.11 rilasciato	È disponibile la versione 2.0.11 del componente di avvio Lambda. Questa versione supporta Lambda Manager 2.3.0.	15 settembre 2023
È stato rilasciato il broker Moquette MQTT 3.1.1 v2.3.4	È disponibile il componente broker Moquette MQTT 3.1.1 v2.3.4.	1 settembre 2023
Framework di test Greengrass	GTF è una raccolta di elementi costitutivi per supportare end-to-end l'automazione. Consente ai clienti AWS IoT Greengrass Version 2 interni di utilizzare lo stesso framework di test utilizzat o dal team di assistenza per le modifiche qualificanti del software, l'accettazione automatica e la garanzia della qualità.	11 agosto 2023
AWS IoT Greengrass Aggiornamento del software Core v2.11.2	Questa versione fornisce la versione 2.11.2 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	9 agosto 2023

Kit di sviluppo Greengrass CLI v1.3.0	È disponibile la versione 1.3.0 del Greengrass Developme nt Kit CLI. Questa versione aggiunge un nuovo `test-e2e` comando per supportare il end-to-end test dei component i utilizzando Open Test Framework.	21 luglio 2023
AWS IoT Greengrass Aggiornamento del software Core v2.11.1	Questa versione fornisce la versione 2.11.1 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	21 luglio 2023
È stato rilasciato Disk spooler v1.0.0	È disponibile il componente Disk spooler v1.0.0.	28 giugno 2023
AWS IoT Greengrass Aggiornamento del software Core v2.11.0	Questa versione fornisce la versione 2.11.0 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	28 giugno 2023
AWS IoT Greengrass Aggiornamento del software Core v2.10.3	Questa versione fornisce la versione 2.10.3 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	21 giugno 2023
AWS IoT Greengrass Aggiornamento del software Core v2.10.2	Questa versione fornisce la versione 2.10.2 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	5 giugno 2023

AWS IoT Greengrass Aggiornamento del software Core v2.10.1	Questa versione fornisce la versione 2.10.1 del componente Greengrass nucleus e aggiorna i componenti forniti. AWS	11 maggio 2023
AWS IoT Greengrass Aggiornamento del software Core v2.10.0	Questa versione fornisce la versione 2.10.0 del componente Greengrass nucleus e aggiorna i componenti forniti. AWS	9 maggio 2023
SageMaker Edge Manager non è più disponibile	Il componente Amazon SageMaker Edge Manager verrà interrotto il 26 aprile 2024.	28 aprile 2023
AWS IoT Greengrass Aggiornamento del software Core v2.9.6	Questa versione fornisce la versione 2.9.6 del componente Greengrass nucleus e aggiorna i componenti forniti. AWS	20 aprile 2023
Rilasciato Log Manager v2.3.2	Il componente Log Manager v2.3.2 è disponibile.	19 aprile 2023

È stato rilasciato lo Stream Manager v2.1.4	Stream manager v2.1.4 è ora disponibile. Questa versione risolve un problema a causa del quale le voci relative alla stessa risorsa di proprietà con lo stesso timestamp all'interno di un singolo batch vengono restituite `ConflictingOperationException` dall' SiteWise API, il che fa sì che stream manager riprovi continuamente. Questa versione aggiorna anche il timeout di connessione predefinito da 3 secondi a 1 minuto.	13 aprile 2023
Kit di sviluppo Greengrass CLI v1.2.3	È disponibile la versione 1.2.3 del Greengrass Development Kit CLI. Questa versione contiene correzioni di bug.	13 aprile 2023
È stato rilasciato il componente di autenticazione del dispositivo client v2.4.0	È disponibile il componente di autenticazione del dispositivo client v2.4.0. Questa versione aggiunge il supporto per l'autenticazione dei dispositivi client per l'emissione di metriche operative che possono essere visualizzate sulla dashboard del dispositivo client Greengrass.	10 aprile 2023

Kit di sviluppo Greengrass CLI v1.2.2	È disponibile la versione 1.2.2 del Greengrass Development Kit CLI. Questa versione contiene miglioramenti e correzioni di bug.	7 aprile 2023
AWS IoT Greengrass Aggiornamento del software Core v2.9.5	Questa versione fornisce la versione 2.9.5 del componente Greengrass nucleus e aggiorna i componenti forniti. AWS	30 marzo 2023
È stato rilasciato Stream Manager v2.1.3	Stream manager v2.1.3 è ora disponibile. Questa versione corregge un problema di avvio nel sistema operativo Windows quando viene eseguito come utente SYSTEM.	7 marzo 2023
È stato rilasciato l'adattatore di protocollo Modbus-RTU v2.1.5	È disponibile il componente dell'adattatore di protocollo Modbus-RTU v2.1.5. Questa versione corregge un problema con l'operazione. `ReadDiscreteInput`	7 marzo 2023

È stato rilasciato il componente di autenticazione del dispositivo client v2.3.2	È disponibile il componente di autenticazione del dispositivo client v2.3.2. Questa versione aggiunge il supporto per la memorizzazione nella cache delle informazioni sul nome host in modo che il componente generi correttamente gli oggetti del certificato quando viene riavviato in modalità offline.	7 marzo 2023
AWS IoT Device Tester la versione 4.7.0 supporta la versione 2.9.4 di Greengrass nucleus	La versione 4.7.0 di IDT per AWS IoT Greengrass V2 ora supporta la versione 2.9.4 di Greengrass nucleus.	2 marzo 2023
Rilasciata l'interfaccia a riga di comando Greengrass v1.2.0	È disponibile l'interfaccia a riga di comando Greengrass v1.2.0.	28 febbraio 2023
AWS IoT Greengrass Aggiornamento del software Core v2.9.4	Questa versione fornisce la versione 2.9.4 del componente Greengrass nucleus e aggiorna i componenti forniti. AWS	24 febbraio 2023

È stato rilasciato Shadow Manager v2.3.1	Shadow manager v2.3.1 è disponibile. Questa versione corregge una condizione che potrebbe impedire la sincroniz zazione degli aggiornamenti di Cloud Shadow. Questa versione corregge anche un problema per cui le modifiche alla configurazione di Named Shadow Sync si applicano a una sola shadow denominata.	21 febbraio 2023
AWS IoT Device Tester la versione 4.7.0 supporta la versione 2.9.3 di Greengrass nucleus	La versione 4.7.0 di IDT per AWS IoT Greengrass V2 ora supporta la versione 2.9.3 di Greengrass nucleus.	9 febbraio 2023
Le migliori pratiche IAM sono state aggiornate	Guida aggiornata per l'allinea mento alle best practice IAM. Per ulteriori informazioni, consulta Best practice per la sicurezza in IAM.	3 febbraio 2023
AWS IoT Greengrass Aggiornamento del software Core v2.9.3	Questa versione fornisce la versione 2.9.3 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	1 febbraio 2023
Rilasciato Log Manager v2.3.1	È disponibile il gestore di log v2.3.1.	27 gennaio 2023
AWS IoT Device Tester la versione 4.7.0 supporta la versione 2.9.2 di Greengrass nucleus	La versione 4.7.0 di IDT per AWS IoT Greengrass V2 ora supporta la versione 2.9.2 di Greengrass nucleus.	3 gennaio 2023

È stato rilasciato Shadow Manager v2.3.0	Shadow manager v2.3.0 è disponibile. Questa versione corregge un problema che potrebbe impedire la sincroniz zazione delle ombre quando un dispositivo archivia la chiave privata del dispositivo Greengrass in un modulo di sicurezza hardware.	29 dicembre 2022
AWS IoT È stato rilasciato il plugin fleet provisioning v1.2.0	AWS IoT è disponibile il plugin per il provisioning della flotta v1.2.0. Questa versione aggiunge il supporto per il provisioning dei dispositivi tramite richiesta di firma del certificato con percorso di chiave privata configurabile.	22 dicembre 2022
AWS IoT Greengrass Aggiornamento del software Core v2.9.2	Questa versione fornisce la versione 2.9.2 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	22 dicembre 2022
AWS IoT Device Tester è stata rilasciata la versione 4.7.0 con GGV2Q v2.5.0	È disponibile la versione 4.7.0 di IDT per V2. AWS IoT Greengrass Questa versione include la suite di qualifica zione AWS IoT Greengras s V2 (GGV2Q) v2.5.0 e supporta le versioni 2.9.1, 2.9.0, 2.8.1, 2.8.0, 2.7.0 e 2.6.0 di Greengrass nucleus.	13 dicembre 2022

È stato rilasciato Shadow Manager v2.2.4	Risolve un problema per cui la convalida delle dimensioni dell'ombra non era coerente con quella del cloud durante l'aggiorn amento del documento shadow locale. Questo risolve anche un problema per cui lo shadow manager interromp e l'ascolto degli aggiornam enti di configurazione se una distribuzione esegue una operazione RESET sui nodi di configurazione.	8 dicembre 2022
Rilasciato Lookout for Vision Edge Agent 1.1.1	È disponibile il componente Lookout for Vision Edge Agent v1.1.1.	5 dicembre 2022
È stato rilasciato Log Manager v2.3.0	Il componente Log Manager v2.3.0 è disponibile.	18 novembre 2022
AWS IoT Device Tester v4.5.11 supporta la versione 2.9.1 di Greengrass nucleus	La versione 4.5.11 di IDT per AWS IoT Greengrass V2 ora supporta la versione 2.9.1 di Greengrass nucleus.	18 novembre 2022
AWS IoT Greengrass Aggiornamento del software Core v2.9.1	Questa versione fornisce la versione 2.9.1 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	18 novembre 2022
AWS IoT Device Tester v4.5.11 supporta la versione 2.9.0 di Greengrass nucleus	La versione 4.5.11 di IDT per AWS IoT Greengrass V2 ora supporta la versione 2.9.0 di Greengrass nucleus.	17 novembre 2022

È stato rilasciato lo Stream Manager v2.1.2	Stream manager v2.1.2 è ora disponibile. Questa versione corregge un problema relativo al sistema operativo Windows che utilizza una lingua diversa dall'inglese.	15 novembre 2022
AWS IoT Greengrass Aggiornamento del software Core v2.9.0	Questa versione fornisce la versione 2.9.0 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	15 novembre 2022
AWS IoT Device Tester v4.5.11 supporta la versione 2.8.1 di Greengrass nucleus	La versione 4.5.11 di IDT per AWS IoT Greengrass V2 ora supporta la versione 2.8.1 di Greengrass nucleus.	19 ottobre 2022
AWS IoT Device Tester È stata rilasciata la versione 4.5.11 con GGV2Q v2.4.1	È disponibile la versione 4.5.11 di IDT per V2. AWS IoT Greengrass Questa versione include la suite di qualifica zione AWS IoT Greengrass V2 (GGV2Q) v2.4.1 e supporta le versioni 2.8.0, 2.7.0 e 2.6.0 di Greengrass nucleus.	13 ottobre 2022
AWS IoT Greengrass Aggiornamento del software Core v2.8.1	Questa versione fornisce la versione 2.8.1 del component e Greengrass nucleus e AWS aggiorna i componenti forniti.	13 ottobre 2022
AWS IoT Greengrass Aggiornamento del software Core v2.8.0	Questa versione fornisce la versione 2.8.0 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS	7 ottobre 2022

AWS CloudFormation È stato aggiunto il supporto per le implementazioni	AWS CloudFormation ora supporta le AWS IoT Greengrass distribuzioni come risorsa.	6 ottobre 2022
SageMaker Rilasciato Edge Manager v1.3.0	Il componente Amazon SageMaker Edge Manager v1.3.0 è disponibile. Questa versione aggiunge il supporto per questo componente per impostare la dimensione del disco per la cache del modello TensorRT e migliora la concorrenza di previsione per utilizzare meglio i motori di accelerazione dei dispositivi come le GPU.	1 settembre 2022
Usa il client di comunicazione tra processi (IPC) V2	Sono state aggiunte informazioni sul client IPC V2, che riducono la quantità di codice da scrivere per utilizzare le operazioni IPC e aiutano a evitare gli errori comuni che possono verificarsi con il client IPC V1.	12 agosto 2022
AWS IoT Device Tester è stata rilasciata la versione 4.5.8 con GGV2Q v2.4.0	È disponibile la versione 4.5.8 di IDT per V2. AWS IoT Greengrass Questa versione include la suite di qualificazione AWS IoT Greengrass V2 (GGV2Q) v2.4.0 e supporta le versioni 2.7.0, 2.6.0 e 2.5.6 di Greengrass nucleus.	12 agosto 2022

SageMaker È stato rilasciato Edge Manager v1.2.0	Il componente Amazon SageMaker Edge Manager v1.2.0 è disponibile. Questa versione aggiunge il supporto per questo componente per recuperare automatic amente i modelli SageMaker NEO-compilati che carichi su Amazon S3, in modo da poter distribuire nuovi modelli senza dover creare una distribuzione. AWS IoT Greengrass	3 agosto 2022
AWS IoT Device Tester v4.5.3 supporta la versione 2.7.0 di Greengrass nucleus	La versione 4.5.3 di IDT per AWS IoT Greengrass V2 ora supporta la versione 2.7.0 di Greengrass nucleus.	1 agosto 2022
È stato rilasciato lo Stream Manager v2.1.0	Stream manager v2.1.0 è ora disponibile. Questa versione include il supporto per l'invio di metriche di telemetria ad Amazon. EventBridge	28 luglio 2022
AWS IoT Greengrass Aggiornamento del software Core v2.7.0	Questa versione fornisce la versione 2.7.0 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS Include il supporto per l'invio di metriche di telemetria ad Amazon. EventBridge	28 luglio 2022

Rilasciato il SiteWise publisher IoT v2.2.0	È disponibile il componente IoT SiteWise Publisher v2.2.0. Questa versione aggiorna il componente per comprimere i dati prima di inviarli al AWS IoT SiteWise servizio, il che riduce l'utilizzo della larghezza di banda fino al 75%.	19 luglio 2022
Tutorial: Sviluppa un componente che interagisca con le ombre dei dispositivi client	È stato aggiunto un nuovo modulo al Tutorial: Interagisci con i dispositivi IoT locali tramite MQTT che puoi seguire per imparare a sviluppare un componente che interagisce con le ombre dei dispositivi client.	18 luglio 2022
Scegli un broker MQTT locale	Sono state aggiunte informazioni su come scegliere un broker MQTT locale in cui i dispositivi client si connettono a un dispositivo principale.	18 luglio 2022
AWS IoT Device Tester v4.5.3 supporta la versione 2.6.0 di Greengrass nucleus	La versione 4.5.3 di IDT per AWS IoT Greengrass V2 ora supporta la versione 2.6.0 di Greengrass nucleus.	29 giugno 2022

AWS IoT Greengrass Aggiornamento del software Core v2.6.0	Questa versione fornisce la versione 2.6.0 del componente Greengrass nucleus e aggiorna i componenti forniti. AWS Include il supporto per gli shadow dei dispositivi client e un broker MQTT 5 locale per i dispositivi client. Include anche il supporto per i caratteri jolly negli argomenti di pubblicazione/sottoscrizione locali, le variabili di ricetta nelle configurazioni dei componenti e i caratteri jolly nelle politiche di autorizzazione IPC. Queste funzionalità consentono di sviluppare e configurare più facilmente componenti da distribuire su flotte di dispositivi principali. Questa versione include anche il supporto per i componenti che utilizzano le operazioni IPC che gestiscono le distribuzioni e i componenti locali su un dispositivo principale.	27 giugno 2022

Aggiornamenti dei componenti dei dispositivi client	Sono disponibili l'autenticazione del dispositivo client v2.1.0, il broker MQTT (Moquette) v2.1.0, il bridge MQTT v2.1.1 e il rilevatore IP v2.1.2. Questa versione migliora la rotazione dei certificati, migliora le prestazioni del broker MQTT e risolve i problemi relativi al modo in cui questi componenti gestiscono gli aggiornamenti di ripristino della configurazione.	14 giugno 2022
AWS IoT Device Tester v4.5.3 supporta la versione 2.5.6 di Greengrass nucleus	La versione 4.5.3 di IDT per AWS IoT Greengrass V2 ora supporta la versione 2.5.6 di Greengrass nucleus.	1 giugno 2022
AWS IoT Greengrass Aggiornamento del software Core v2.5.6	Questa versione fornisce la versione 2.5.6 del componente Greengrass nucleus e aggiorna i componenti forniti. AWS Include il supporto per i moduli di sicurezza hardware con chiavi ECC. Include anche altre correzioni di bug e miglioramenti.	31 maggio 2022
AWS IoT Rilasciato il plugin fleet provisioning v1.1.0	AWS IoT è disponibile il plugin fleet provisioning v1.1.0. Questa versione aggiunge il supporto per formati di percorsi di file aggiuntivi quando si configura il plug-in su dispositivi Windows.	12 maggio 2022

Rilasciati nuovi runtime Lambda	È stato aggiunto il supporto per i nuovi runtime Lambda: Python 3.9, Java 11 e NodeJS 14.	10 maggio 2022
Sviluppa un componente Greengrass che rinvii gli aggiornamenti dei componenti	È stato aggiunto un tutorial che puoi seguire per imparare a sviluppare un component e Greengrass che differisc e gli aggiornamenti dei componenti dalle distribuz ioni. Ad esempio, potresti voler ritardare un aggiornam ento quando un dispositivo ha un livello di batteria basso o mentre è in esecuzione un processo che non può essere interrotto.	4 maggio 2022
CloudWatch sono state rilasciate le versioni 3.1.0 e 3.1.0 di metrics AWS IoT Device Defender	CloudWatch sono disponibi li il componente metrics v3.1.0 e il componente v3.1.0. AWS IoT Device Defender Queste versioni aggiungon o il supporto per le configura zioni proxy di rete HTTPS. Per ulteriori informazioni, consulta Connect sulla porta 443 o tramite un proxy di rete e Abilita il dispositivo principal e a considerare attendibile un proxy HTTPS.	27 aprile 2022
Esegui la migrazione da AWS IoT Greengrass Version 1	È stata aggiunta una guida che puoi seguire per migrare da AWS IoT Greengrass V1 a. AWS IoT Greengrass V2	26 aprile 2022

AWS IoT Device Tester v4.5.3 con GGV2Q v2.3.1 aggiornato e IDT v4.5.1 con GGV2Q v2.3.0 aggiunto alle versioni supportate	La versione 4.5.3 di IDT per AWS IoT Greengrass V2 con suite di qualificazione V2 (AWS IoT Greengrass GGV2Q) v2.3.1 è stata aggiornata per includere il supporto per le versioni 2.5.5, 2.5.4 e 2.5.3 di Greengrass nucleus. Questo aggiornamento include anche IDT 4.5.1 con suite di qualificazione AWS IoT Greengrass V2 (GGV2Q) v2.3.0 come versione supportata. IDT 4.5.1 con suite di qualificazione AWS IoT Greengrass V2 (GGV2Q) v2.3.0 supporta la versione 2.5.3 di Greengrass nucleus.	25 aprile 2022
È stato rilasciato l'adattatore di protocollo Modbus-RTU v2.1.0	È disponibile il componente dell'adattatore di protocollo Modbus-RTU v2.1.0. Questa versione aggiunge nuovi parametri che è possibile specificare per configurare la comunicazione seriale con i dispositivi Modbus RTU.	20 aprile 2022

CloudWatch rilasciati metrics v2.1.0, Firehose v2.1.0 e Amazon SNS v2.1.0	CloudWatch sono disponibili il componente metrics v2.1.0, il componente Firehose v2.1.0 e il componente Amazon SNS v2.1.0. Queste versioni aggiungono il supporto per le configurazioni proxy di rete HTTPS. Per ulteriori informazioni, consulta Connect sulla porta 443 o tramite un proxy di rete e Abilita il dispositivo principale a considerare attendibile un proxy HTTPS.	19 aprile 2022
AWS IoT Device Tester è stata rilasciata la versione 4.5.3 con GGV2Q v2.3.1	È disponibile la versione 4.5.3 di IDT per V2. AWS IoT Greengrass Questa versione include la suite di qualificazione AWS IoT Greengrass V2 (GGV2Q) v2.3.1 e supporta la versione 2.5.5 di Greengrass nucleus.	15 aprile 2022

AWS IoT Greengrass Aggiornamento del software Core v2.5.5	Questa versione fornisce la versione 2.5.5 del componente Greengrass nucleus e aggiorna i componenti forniti. AWS Aggiunge il supporto per i dispositivi Windows che utilizzano una lingua di visualizzazione diversa dall'inglese. Risolve inoltre un problema per cui il dispositivo principale non segnalava il proprio stato al servizio AWS IoT Greengrass cloud dopo il provisioning in determinati scenari.	6 aprile 2022
AWS IoT Greengrass Aggiornamento del software Core v2.5.4	Questa versione fornisce la versione 2.5.4 del componente Greengrass nucleus e aggiorna i componenti forniti. AWS Include correzioni di bug e miglioramenti.	23 marzo 2022
Scarica AWS IoT Device Tester a livello di codice	Sono state aggiunte informazioni su come scaricare IDT a livello di codice. AWS IoT Greengrass V2	15 marzo 2022

Kit di sviluppo Greengrass CLI v1.1.0	È disponibile la versione 1.1.0 del Greengrass Developme nt Kit CLI. Questa versione aggiunge nuovi argomenti ai comandi and. `component init` `component publish` Questa versione aggiorna anche il `component publish` comando per creare il componente se non è compilato.	24 febbraio 2022
È stato rilasciato Shadow Manager v2.1.0	È disponibile il component e Shadow Manager v2.1.0. Questa versione aggiunge l'opzione per configura re l'intervallo con cui il componente sincronizza le ombre. AWS IoT Core Ad esempio, è possibile specifica re un intervallo più lungo per ridurre l'utilizzo della larghezza di banda e i costi.	3 febbraio 2022
Immagini Dockerfile e Docker per il software Core v2.5.3 AWS IoT Greengrass	Le immagini Dockerfile e Docker per il software Core v2.5.3 sono ora disponibili. AWS IoT Greengrass	12 gennaio 2022

AWS IoT Device Tester è stata rilasciata la versione 4.5.1 con GGV2Q v2.3.0	È disponibile la versione 4.5.1 di IDT per V2. AWS IoT Greengrass Questa versione include la suite di qualifica zione AWS IoT Greengrass V2 (GGV2Q) v2.3.0 e supporta la convalida e la qualifica zione dei dispositivi basati su Linux che utilizzano un modulo di sicurezza hardware (HSM) per archiviare la chiave privata e il certificato utilizzat i dal software Core. AWS IoT Greengrass	11 gennaio 2022
AWS IoT Greengrass Aggiornamento del software Core v2.5.3	Questa versione fornisce la versione 2.5.3 del component e Greengrass nucleus e aggiorna i componenti forniti. AWS Include il supporto per configurare il software AWS IoT Greengrass Core in modo che utilizzi una chiave privata e un certificato da archiviare in modo sicuro in un modulo di sicurezza hardware (HSM).	6 gennaio 2022
Immagini Dockerfile e Docker per il software Core v2.5.2 AWS IoT Greengrass	Le immagini Dockerfile e Docker per il software Core v2.5.2 sono ora disponibili. AWS IoT Greengrass	20 dicembre 2021

AWS IoT Device Tester è stata rilasciata la versione 4.4.1 con GGV2Q v2.2.1	È disponibile la versione 4.4.1 di IDT per V2. AWS IoT Greengrass Questa versione include la suite di qualifica zione AWS IoT Greengrass V2 (GGV2Q) v2.2.1 e supporta la versione 2.5.2 di Greengrass nucleus per la qualificazione dei dispositivi.	12 dicembre 2021
Esegui inferenze di apprendim ento automatico utilizzando Amazon Lookout for Vision	Sono state aggiunte informazi oni su come eseguire l'inferen za dell'apprendimento automatico utilizzando Lookout for Vision sui dispositivi core Greengrass. Lookout for Vision utilizza la visione artificiale per individuare difetti visivi nei prodotti industriali.	8 dicembre 2021
AWS IoT Device Tester è stata rilasciata la versione 4.4.1 con GGV2Q v2.2.0	È disponibile la versione 4.4.1 di IDT per V2. AWS IoT Greengrass Questa versione include la suite di qualifica zione AWS IoT Greengrass V2 (GGV2Q) v2.2.0 e supporta la versione 2.5.2 di Greengrass nucleus per la qualificazione dei dispositivi.	6 dicembre 2021

AWS IoT Greengrass Aggiornamento del software Core v2.5.2	Questa versione fornisce la versione 2.5.2 del componente Greengrass nucleus e aggiorna i componenti forniti. AWS Risolve un problema con il servizio Windows che si verifica dopo l'aggiornamento del nucleo Greengrass. Include anche il supporto per il AWS IoT Device Defender componente sui dispositivi Windows.	3 dicembre 2021
Nuovo connettore edge per il componente Kinesis Video Streams	È disponibile la versione 1.0.0 del connettore perimetrale per il componente Kinesis Video Streams. Questo AWS-provided legge i feed video delle telecamere locali e pubblica i flussi su Kinesis Video Streams. Questo componente si integra con AWS IoT TwinMaker, il che consente di visualizzare e gestire flussi video e altri dati nelle dashboard di Grafana.	30 novembre 2021
Gestisci i dispositivi core Greengrass con AWS Systems Manager	Sono state aggiunte informazioni su come gestire i dispositivi core Greengrass con. AWS Systems Manager Systems Manager è un AWS servizio che consente di visualizzare i dati operativi, automatizzare le attività operative e mantenere la sicurezza e la conformità.	29 novembre 2021

CLI del kit di sviluppo Greengrass	Sono state aggiunte informazioni sulla AWS IoT Greengrass Development Kit Command-Line Interface (GDK CLI), uno strumento che puoi scaricare sul tuo computer di sviluppo locale per aiutarti a sviluppare componenti Greengrass personalizzati. Puoi usare la CLI GDK per creare, creare e pubblicare componenti personalizzati.	29 novembre 2021
Componenti Greengrass forniti dalla community	Sono state aggiunte informazioni sul Greengrass Software Catalog, che è un indice dei componenti Greengrass sviluppati dalla comunità Greengrass. Da questo catalogo, puoi scaricare, modificare e distribuire componenti per creare le tue applicazioni Greengrass.	29 novembre 2021
AWS IoT Greengrass Aggiornamento del software Core v2.5.1	Questa versione fornisce la versione 2.5.1 del componente Greengrass nucleus e AWS aggiorna i componenti forniti. Include il supporto per Java a 32 bit su dispositivi Windows. Risolve inoltre i problemi relativi al nuovo comportamento di rimozione dei gruppi di oggetti e al caricamento delle variabili di ambiente di sistema sui dispositivi Windows.	23 novembre 2021

AWS IoT Device Tester è stata rilasciata la versione 4.4.0 con GGV2Q v2.1.0	È disponibile la versione 4.4.0 di IDT per V2. AWS IoT Greengrass Questa versione include la suite di qualifica zione AWS IoT Greengrass V2 (GGV2Q) v2.1.0 e supporta la qualificazione dei dispositi vi Greengrass basati su Windows che eseguono la versione 2.5.0 di Greengrass nucleus.	19 novembre 2021
AWS IoT Greengrass Aggiornamento del software Core v2.5.0	Questa versione fornisce la versione 2.5.0 del component e Greengrass nucleus e AWS aggiorna i component i forniti. Include il supporto per l'esecuzione del software AWS IoT Greengrass Core su dispositivi Windows. Inoltre, modifica il comportamento di rimozione dei gruppi di oggetti e aggiunge il supporto per i proxy HTTPS.	12 novembre 2021

SageMaker Rilasciato Edge Manager v1.1.0	Il componente Amazon SageMaker Edge Manager v1.1.0 è disponibile. Questa versione aggiunge il supporto per i dispositivi core Greengrass che eseguono Amazon Linux 2 e aggiunge un nuovo parametro di configurazione per specifica re la posizione della cartella dei dati di acquisizione sul dispositivo.	3 novembre 2021
Aggiornamento della prevenzione del "confused deputy" tra servizi	AWS IoT Greengras s V2 supporta l'utilizzo delle chiavi di contesto `aws:SourceArn` e della condizione `aws:Sourc eAccount` globale nelle politiche delle risorse IAM per prevenire il confuso problema del vice.	1° novembre 2021
aggiornamenti dei componenti dei dispositivi client	Sono disponibili l'autentica zione del dispositivo client v2.0.3, il rilevatore IP v2.1.0, il bridge MQTT v2.1.0 e il broker MQTT (Moquette) v2.0.2. Questa versione aggiunge il supporto completo per le porte del broker MQTT non predefini te e include altre correzioni di bug e miglioramenti.	28 ottobre 2021

È stato rilasciato Shadow Manager v2.0.4	Il componente Shadow Manager v2.0.4 è disponibile. Questa versione corregge un problema che causava l'eliminazione di versioni appena create di qualsiasi shadow precedentemente eliminata da shadow manager. A partire da questa versione, l'operazione `DeleteThingShadow` IPC incrementa la versione shadow.	20 ottobre 2021
Rilasciato Log Manager v2.2.0	Il componente Log Manager v2.2.0 è disponibile. Il gestore dei registri ora supporta l'utilizzo di una mappa di configurazione per fornire configurazioni di registro dei componenti.	20 ottobre 2021
Lambda Manager v2.1.4 rilasciato	È disponibile il componente Lambda Manager v2.1.4. Questa versione corregge un problema che causava l'elaborazione di un solo messaggio da parte delle funzioni Lambda che utilizzano i runtime NodeJS.	20 ottobre 2021
Usa la comunicazione tra processi, AWS le credenziali e lo stream manager nei componenti del contenitore Docker	Sono state aggiunte informazioni su come utilizzare la comunicazione tra processi (IPC), AWS le credenziali e lo stream manager nei componenti personalizzati del contenitore Docker.	19 ottobre 2021

Nuovo componente dell'emettitore di telemetria del nucleo	È disponibile la versione 1.0.0 del componente emettitore di telemetria a nucleo. Questo componente AWS fornito raccoglie dati di telemetria sullo stato del sistema e li pubblica continuamente su un argomento locale e su un argomento MQTT. AWS IoT Core	30 settembre 2021
Consenti il traffico dei dispositivi attraverso un proxy o un firewall	Sono state aggiunte informazioni sugli endpoint e sulle porte utilizzate dai dispositivi core Greengrass, in modo da poter limitare il traffico come misura di sicurezza.	16 settembre 2021
AWS IoT Device Tester è stata rilasciata la versione 4.2.0 con GGV2Q v2.0.1	La versione 4.2.0 di IDT per AWS IoT Greengrass V2 è stata aggiornata con la suite di qualificazione V2 (GGV2Q) v2.0.1. AWS IoT Greengrass Questa versione supporta la versione 2.4.0 di Greengrass nucleus per la qualificazione dei dispositivi.	31 agosto 2021

Componenti di installazione aggiornati per l'apprendimento automatico	Sono disponibili il componente di installazione DLR v1.6.5 e il componente di installazione TensorFlow Lite v2.5.4. Queste versioni dei componenti includono il nuovo parametro di `UseInstaller` configurazione che consente di disabilitare lo script di installazione predefinito.	30 agosto 2021
Supporto Linux integrato per AWS IoT Greengrass	La BitBake ricetta per AWS IoT Greengrass V2 è disponibile nel `meta-aws` progetto su GitHub. È possibile utilizzare questa ricetta per creare un sistema operativo personalizzato basato su Linux utilizzando il progetto Yocto.	20 agosto 2021
Integrità del codice	Sono state aggiunte informazioni su come AWS IoT Greengrass V2 verifica l'integrità del software che i dispositivi core Greengrass scaricano da. Cloud AWS	19 agosto 2021
Endpoint VPC ()AWS PrivateLink	AWS IoT Greengrass ora supporta l'interfaccia VPC endpoints (AWS PrivateLink) per il AWS IoT Greengrass piano di controllo. Puoi stabilire una connessione privata tra il tuo VPC e il piano di AWS IoT Greengrass controllo.	16 agosto 2021

È stato rilasciato lo Stream Manager v2.0.12	Stream manager v2.0.12 è ora disponibile. Questa versione corregge un problema che impediva gli aggiornam enti dalla versione 2.0.7 del componente stream manager a una versione compresa tra v2.0.8 e v2.0.11.	10 agosto 2021
Immagini AWS IoT Greengras s Dockerfile e Docker per il software Core v2.4.0	Le immagini Dockerfile e Docker per il software Core v2.4.0 sono ora disponibili. AWS IoT Greengrass	9 agosto 2021
AWS IoT Greengrass Aggiornamento del software Core v2.4.0	Questa versione fornisce la versione 2.4.0 del component e Greengrass nucleus e AWS aggiorna i componenti forniti. Include il supporto per i limiti delle risorse del sistema di componenti, le operazioni IPC per sospendere e riprender e i componenti e i plug-in di provisioning.	3 agosto 2021
AWS IoT SiteWise Nuovi componenti	Sono stati aggiunti i seguenti componenti AWS forniti per AWS IoT SiteWise: raccoglit ore IoT SiteWise OPC-UA, editoreIoT e SiteWise processore IoT. SiteWise	29 luglio 2021

AWS IoT Device Tester è stata rilasciata la versione 4.2.0 con GGV2Q v2.0.0	È disponibile la versione 4.2.0 di IDT per V2. AWS IoT Greengrass Questa versione include la suite di qualifica zione AWS IoT Greengras s V2 (GGV2Q) v2.0.0 e include il supporto per test di qualificazione opzionali per i componenti Docker, l'apprend imento automatico e lo stream manager.	14 luglio 2021
AWS IoT GreengrassSDK per dispositivi AWS IoT Libreria IPC di base disponibile in C++ v2	La versione 1.13.0 di SDK per dispositivi AWS IoT for C++ v2 supporta AWS IoT Greengras s Core IPC, quindi è possibile sviluppare componenti in C ++ che interagiscono con il software Core. AWS IoT Greengrass	14 luglio 2021
SageMaker È stato rilasciato il componente Edge Manager v1.0.2	Il componente Amazon SageMaker Edge Manager v1.0.2 è disponibile. Questa versione aggiorna lo script di installazione nel ciclo di vita del componente. I tuoi dispositi vi principali devono ora avere Python 3.6 o versione successiva, inclusa `pip` la tua versione di Python, installat o sul dispositivo prima di distribuire questo componente.	12 luglio 2021

Aggiornamento di supporto AWS IoT Device Tester per la AWS IoT Greengrass versione 2	IDT per AWS IoT Greengrass V2 versione 4.1.0 ora supporta l'utilizzo della versione 2.3.0 di Greengrass nucleus per la qualificazione dei dispositivi.	8 luglio 2021
Immagini Dockerfile e Docker per il software Core v2.3.0 AWS IoT Greengrass	Le immagini Dockerfile e Docker per il software Core v2.3.0 sono ora disponibili. AWS IoT Greengrass	7 luglio 2021
AWS politiche gestite	Sono state aggiunte informazioni sulle politiche AWS gestite per AWS IoT Greengrass.	2 luglio 2021
Nuove opzioni JVM consigliate	Sono state aggiunte informazioni sulle opzioni JVM consigliate per controllare l'allocazione della memoria per il software Core. AWS IoT Greengrass	30 giugno 2021
AWS IoT Greengrass Aggiornamento del software Core v2.3.0	Questa versione fornisce la versione 2.3.0 del componente Greengrass nucleus e AWS aggiorna i componenti forniti. Include il supporto per documenti di configurazione di componenti di grandi dimensioni nelle distribuzioni.	29 giugno 2021
Immagini Dockerfile e Docker per il software Core v2.2.0 AWS IoT Greengrass	Le immagini Dockerfile e Docker per il software Core v2.2.0 sono ora disponibili. AWS IoT Greengrass	28 giugno 2021

AWS IoT Device Tester è stata rilasciata la versione 4.1.0 con GGV2Q v1.1.1	È disponibile la versione 4.1.0 di IDT per V2. AWS IoT Greengrass Questa versione include la suite di qualifica zione AWS IoT Greengrass V2 (GGV2Q) v1.1.1 e supporta l'utilizzo di Greengrass nucleus v2.2.0, v2.1.0 e v2.0.5 per la qualificazione dei dispositivi.	18 giugno 2021
AWS IoT Greengrass Aggiornamento del software Core v2.2.0	Questa versione fornisce la versione 2.2.0 del component e Greengrass nucleus e AWS aggiorna i component i forniti. Include componenti che è possibile distribuire per aggiungere il supporto per i dispositivi client e aggiungere il servizio shadow locale.	18 giugno 2021
Lambda Launcher v2.0.6 rilasciato	È disponibile la versione 2.0.6 del componente di avvio Lambda. Questa versione include miglioramenti delle prestazioni e correzioni di bug.	13 giugno 2021
Rilasciato il nuovo component e SageMaker Edge Manager	La versione 1.0.0 del componente Amazon SageMaker Edge Manager è disponibile per. AWS IoT Greengrass Questo componente installa il binario dell'agente SageMaker Edge Manager sui dispositivi core Greengrass.	10 giugno 2021

Tipi di componenti	Sono state aggiunte informazioni sui tipi di componenti in AWS IoT Greengrass. Il tipo di componente specifica in che modo il software AWS IoT Greengrass Core esegue un componente.	3 giugno 2021
AWS IoT Device Tester è stata rilasciata la versione 4.0.2 con la versione 1.1.0 di GGV2Q	È disponibile la versione 4.0.2 di IDT per V2. AWS IoT Greengrass Questa versione include la suite di qualificazione AWS IoT Greengrass V2 (GGV2Q) v1.1.0 e supporta l'utilizzo di Greengrass nucleus v2.1.0 con Greengrass CLI v2.1.0 per la qualificazione dei dispositivi. Ciò include anche nuovi gruppi di test obbligatori per MQTT e Lambda e altre correzioni di bug e miglioramenti minori.	5 maggio 2021
Immagini Dockerfile e Docker per il software Core v2.1.0 AWS IoT Greengrass	Le immagini Dockerfile e Docker per il software Core v2.1.0 sono ora disponibili. AWS IoT Greengrass L'immagine Docker consente di eseguire il software AWS IoT Greengrass Core in un contenitore Docker che utilizza Amazon Linux 2 come sistema operativo di base.	27 aprile 2021

AWS IoT Greengrass Aggiornamento del software Core v2.1.0	Questa versione fornisce la versione 2.1.0 del component e Greengrass nucleus e AWS aggiorna i componenti forniti. Include un nuovo component e che puoi utilizzare per scaricare immagini Docker da repository Amazon ECR privati e nuovi component i di esempio per eseguire inferenze di machine learning utilizzando Lite. TensorFlow	26 Aprile 2021
Componente di esempio che utilizza Secrets Manager	È stato aggiunto un componente di esempio che stampa il valore di un AWS Secrets Manager segreto distribuito su un dispositivo principale.	8 aprile 2021
AWS IoT Politica minima per i dispositivi core Greengrass	Sono state aggiunte informazioni sul set minimo di autorizzazioni necessarie per supportare le funzionalità di base di Greengrass su un dispositivo principale.	2 aprile 2021
Iscriviti agli stream di eventi IPC	Sono state aggiunte informazioni su come utilizzare le operazioni di comunicazione tra processi (IPC) per sottoscrivere flussi di eventi su un dispositivo core Greengrass.	1 aprile 2021

Aggiornamento di supporto AWS IoT Device Tester per AWS IoT Greengrass	IDT per AWS IoT Greengrass V2 versione 4.0.1 ora supporta l'utilizzo della versione 2.0.5 di Greengrass nucleus con la versione 2.0.5 della CLI di Greengrass per la qualifica zione dei dispositivi.	17 marzo 2021
Crea componenti personali zzati che utilizzano stream manager	Sono state aggiunte informazi oni su come configurare le ricette e gli artefatti dei componenti per sviluppare applicazioni che gestiscono i flussi di dati.	9 marzo 2021
AWS IoT Greengrass Aggiornamento del software Core v2.0.5	Questa versione fornisce la versione 2.0.5 del component e Greengrass nucleus e AWS aggiorna i componenti forniti. Risolve un problema relativo al supporto proxy di rete e un problema con l'endpoint AWS del piano dati Greengras s nelle regioni cinesi.	9 marzo 2021
Riferimento alla variabile di ambiente del componente	Sono state aggiunte informazi oni sulle variabili di ambiente che il software AWS IoT Greengrass Core imposta per i componenti. È possibile utilizzare queste variabili di ambiente per ottenere il nome dell'oggetto e la Regione AWS versione del nucleo di Greengrass.	23 febbraio 2021

Installazione manuale	Sono state aggiunte informazioni su come creare AWS le risorse necessarie manualmente o installarle dietro un firewall o un proxy di rete. Utilizzando un'installazione manuale, non è necessario concedere all'installatore l'autorizzazione per creare risorse nel proprio computer Account AWS, in quanto si creano le risorse necessarie AWS IoT e quelle IAM. Puoi anche configurare il dispositivo in modo che si connetta alla porta 443 o tramite un proxy di rete.	17 febbraio 2021
AWS IoT Greengrass Aggiornamento della libreria IPC principale SDK per dispositivi AWS IoT per Python v2	La versione 1.5.4 di SDK per dispositivi AWS IoT for Python v2 semplifica i passaggi necessari per connettersi al servizio Core IPC. AWS IoT Greengrass	11 febbraio 2021
Aggiornamento di supporto AWS IoT Device Tester per AWS IoT Greengrass	IDT per AWS IoT Greengrass V2 versione 4.0.1 ora supporta l'utilizzo della versione 2.0.4 di Greengrass nucleus con la versione 2.0.4 della CLI di Greengrass per la qualificazione dei dispositivi.	5 febbraio 2021

Nuovo tutorial per importare le funzioni Lambda	È stato aggiunto un nuovo tutorial basato su console per importare una funzione Lambda come componente e che viene eseguito sul dispositivo principale Greengrass.	5 febbraio 2021
AWS IoT Greengrass Aggiornamento del software Core v2.0.4	Questa versione fornisce la versione 2.0.4 del componente Greengrass nucleus. Include il nuovo `greengrassDataPlanePort` parametro per configurare la comunicazione HTTPS sulla porta 443 e corregge i bug. La policy IAM minima ora richiede l'indicazione `iam:GetPolicy` e il momento in `sts:GetCallerIdentity` cui viene eseguito il programma di installazione del software AWS IoT Greengrass Core. `--provision true`	4 febbraio 2021
Rilasciato il nuovo componente di tunneling sicuro	La versione 1.0.0 del componente di tunneling sicuro è disponibile per. AWS IoT Greengrass Questo componente AWS fornito utilizza un tunneling AWS IoT sicuro per stabilire una comunicazione bidirezionale sicura con un dispositivo core Greengrass protetto da firewall limitati.	21 gennaio 2021

AWS IoT Device Tester per la versione 4.0.1 rilasciata AWS IoT Greengrass	È disponibile la versione 4.0.1 di IDT per AWS IoT Greengrass V2. Questa versione consente di utilizzare IDT per sviluppare ed eseguire suite di test personalizzate per la convalida dei dispositi vi. Sono incluse anche le applicazioni IDT con firma di codice per macOS e Windows.	22 dicembre 2020
Versione iniziale di AWS IoT Greengrass Version 2	AWS IoT Greengrass V2 è una nuova versione principal e di AWS IoT Greengrass. Questa versione aggiunge diverse funzionalità come componenti software modulari e distribuzioni continue. Queste funzionalità semplific ano lo sviluppo e la gestione di applicazioni edge.	15 dicembre 2020

AWS Glossario

Per la AWS terminologia più recente, consultate il AWS glossario nella sezione Reference.Glossario AWS

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.